VMS

digital

VMS General User's Manual

Order Number: AA-LA98A-TE

VMS General User's Manual

Order Number: AA-LA98A-TE

April 1988

This manual describes general user tasks you can perform using the VMS operating system. The information contained in this manual is intended for all users and is applicable to all members of the VAX and MicroVAX families of computers running the VMS operating system.

Revision/Update Information: This is a new manual.

Software Version: VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	
DECwriter	RSX	

digital™

ZK4323

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA, Puerto Rico, Alaska, and Hawaii call 800-DIGITAL.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript® printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

® PostScript is a trademark of Adobe Systems, Inc.

Contents

Chapter 2 Working with Files and Directories

Chapter 3 Working with Processes

Chapter 4 Using Logical Names

Chapter 5 Representing Data with Symbols

Chapter 6 Writing and Using Command Procedures

Chapter 7 Maintaining Accounts and System Security

Chapter 8 Editing Files with the EVE and EDT Editors

Chapter 9 Processing Files with DIGITAL Standard Runoff

GENERAL USER'S REFERENCE

DCL Commands

Figures

Tables

Preface

This manual provides an overview of the VMS operating system.

Intended Audience

This manual is intended for all users of the VMS operating system.

Document Structure

This manual is organized into two major parts: Concepts and Reference.

The Concepts Section includes the following chapters:

- Chapter 1—Introducing VMS and DCL
- Chapter 2—Working with Files and Directories
- Chapter 3—Working with Processes
- Chapter 4—Using Logical Names
- Chapter 5—Representing Data with Symbols
- Chapter 6—Writing and Using Command Procedures
- Chapter 7—Maintaining Accounts and Security
- Chapter 8—Editing Files with the EVE and EDT Editors
- Chapter 9—Processing Files with DIGITAL Standard Runoff

The Reference Section includes the following reference information and information about general user utilities:

- DCL commands—In alphabetical order, describes all Digital Command Language (DCL) commands and lexical functions.
- DSR commands—Contains a list of rules you must follow when using Digital Standard Runoff to format output and describes all DSR commands in alphabetical order.
- EDT Editor—Provides reference information about EDT keypad editing.

- EVE Commands—In alphabetical order, describes all EVE editing commands that are entered at the *Command:* prompt.
- Mail Utility—Describes the MAIL command and qualifiers that comprise the Mail Utility. The Mail Utility is used to send messages to users on other systems.
- Sort/Merge Utility—Describes the Sort/Merge Utility, which is used to sort records or to merge input files.
- Terminal Fallback Facility—Describes the VMS Terminal Fallback Facility, which provides table-driven character conversion for terminals.

Three appendixes contain the following information:

- ASCII character set
- Expressions
- Terminal Keys

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks (″). The term apostrophe (′) is used to refer to a single quotation mark.

Chapter 1

Introducing VMS and DCL

Your VAX computer operates under the control of the VMS (Virtual Memory System) operating system. VMS is an interactive system: while you are logged in to the computer, you and the system conduct a dialogue of command and response. You use DCL, the DIGITAL Command Language interpreter, to communicate with VMS. DCL provides you with over 200 commands and functions to use in communicating with VMS to accomplish your computing tasks.

This chapter describes basic interaction with the VMS operating system. In it, you will learn how to log in to the system, how to use DCL to communicate with the system, how to customize your computing environment, how to get online help, and how to use two system utilities that let you communicate with other users.

Some of the topics discussed in this chapter require you to be familiar with your terminal. For information on setting up or using your terminal, see the owner's manual supplied with your terminal.

1.1 Logging In to the System

In order to interact with the VMS operating system, you must log in to an account. Logging in consists of identifying yourself to the system as an authorized user. Your system manager or whoever authorizes system use at your installation usually sets up accounts. This person provides you with your user name and password. Your user name is a unique name that identifies you to the system and distinguishes you from other users. In many cases, a user name is your first or last name. Your password is for your protection. If you maintain its secrecy, other users cannot use system resources under your user name.

Use the following procedure to log in to the system:

1. Make sure your terminal is plugged in and the power is turned on.
2. Press RETURN to signal the system that you want to log in. (You may need to press RETURN several times.) The system responds by displaying a prompt for your user name.

3. Enter your user name and press RETURN. The system displays your user name on the screen as you type it. (You have about 30 seconds to do this, otherwise the system "times out" and you must start the login procedure again.) After you enter your user name and press RETURN, the system prompts you for your password.

4. Enter your password and press RETURN. The system does not display your password as you type it; this preserves the secrecy of your password. The password you enter is compared with the encrypted password stored in a system file called the User Authorization File (UAF). (See Section 7.1 for more information about the UAF file.)

If you make a mistake entering your user name or password, or your password has expired, the system displays the message "User authorization failure," and you are not logged in. This message means that you made a typing mistake when entering your user name or password, or that your user name or password is incorrect. If you make a mistake entering your user name or password, press RETURN and try again. If your password has expired or you have any other problems logging in, get help from the person who set up your account.

Some accounts are set up to require two passwords. If you see a second password prompt, enter the second password required to access that account.

If none of these prompts ($, Password:, or Username:) appears when you press RETURN, a system password may be required to log in to your system. If you know the system password, type it and press RETURN. If you do not know it, see the person in charge of your system.

If your login is successful, a dollar sign symbol ($) is displayed in the left margin of your terminal. This dollar sign is a prompt that VMS uses to indicate you are at DIGITAL Command Language (DCL) level and can begin entering DCL commands. When you log in to the system and work interactively with DCL, you are at command level 0. When you execute a program interactively, you are placed at command level 1; when the program completes execution, you are returned to command level 0.

The following example shows a successful login:

```
RET
Username:  CASEY RET
Password:  RET
        Welcome to VAX/VMS version 5.0 on node MARS
    Last interactive login on Friday, 31-DEC-1988 08:41
    Last non-interactive login on Thursday, 30-DEC-1988 11:05
$
```

If your account was set up by someone else, immediately change your password after you log in for the first time. You should also change your password frequently to ensure system security. To change your password, enter the DCL command SET PASSWORD. Enter your old password at the first prompt and press RETURN. Enter your new password at the next prompt and press RETURN. Finally, enter your new

password again and press RETURN to confirm your choice. The following example shows what you see:

```
$ SET PASSWORD
Old password:
New password:
Verification:
```

(If you are managing your own system, see the *VMS System Manager's Manual* for instructions on setting up a user account and establishing a password.)

Each time you log in, the system automatically executes up to two login command procedures. A *command procedure* is a file that contains a list of DCL commands. When a command procedure is executed, the DCL interpreter reads the file and executes the commands it contains.

If your system manager has set up a system login command procedure, it is executed when you log in. This command procedure allows your system manager to ensure that certain commands are always executed when you and other users on your system log in.

After executing the system login command procedure, the system executes your personal login command procedure, if one exists. Your personal login command procedure allows you to customize your computing environment. The commands contained in it are executed every time you log in. The person who set up your account may have placed a login command procedure in your top level directory. (Unless your account has been specially modified to do otherwise, the system automatically places you in your top level directory when you log in.) If a login command procedure is not there, you can create one yourself, name it LOGIN.COM, and place it in your top level directory unless your system manager tells you otherwise. DCL and DCL commands are discussed in Section 1.2. Directories, including your top level directory, are discussed in Chapter 2. A sample personal login command procedure is described in Section 6.3.

When you log in, an environment is created from which you can enter commands. This environment is called your *process*. The system obtains the characteristics that are unique to your process from the user authorization file (UAF). The UAF lists those users permitted to access the system and defines the characteristics for each user's process. The system manager usually maintains the UAF. See Chapter 3 for more information about processes.

1.1.1 Alternative Login Procedures

The standard login procedure described in the preceding section may not fit your needs if you must log in to a terminal assigned to a specific account, access a system other than your own, or dial in to your system by telephone. The following sections describe these procedures.

1.1.1.1 Automatic Login

You may need to log in to a terminal that is assigned to a particular account. This procedure, called automatic login, permits you to log in without specifying a user name. To log in, turn on the terminal and press the RETURN key. Either the DCL or password prompt appears. If the DCL prompt appears, you are logged in. If the password prompt appears, type the password of the account associated with the terminal and press RETURN. (The password is not displayed on the screen.) If your login is successful, you see the DCL dollar sign prompt that VMS uses to indicate you are at DCL command level and can begin entering commands.

1.1.1.2 Logging In Over the Network

Your system may be part of a DECnet–VAX network. VMS systems linked together in a DECnet network are able to communicate with each other and share information and resources. Each system in the network is called a network *node* and is identified by a unique node name and address. When you are logged in to a network node, you can communicate with every other node in the network. The node at which you are logged in is called the *local node*; the other nodes on the network are called *remote nodes*.

If you have access to an account on a remote node, you can log in to that account from your local node and use the facilities of that remote node while remaining physically connected to your local node.

The following example shows how to access a remote node on the network using the DCL command SET HOST. HUBBUB is the name of the remote node.

```
$ SET HOST HUBBUB
```

You can then log in to your account on the remote node using the remote node's login procedure. When you use the SET HOST command to log in to a remote node, you can perform any operation on the remote node as though it were your local node. Note that the remote node need not be a VMS system. If the network link cannot be established, you receive an error message.

If you want to abort the login procedure, enter CTRL/Z at the user name or password prompt or enter CTRL/Y twice. The host system should respond with the question, "Are you repeating ^Y to abort the remote session"? Answering Y (uppercase or lowercase) aborts the remote session.

You can terminate a remote session in two ways:

- Use the remote system's logout procedure (for example, on a VMS system, use the LOGOUT command).
- Press CTRL/Y twice to obtain the host system's prompt that asks whether you want to abort the remote session. Answer Y if you want to abort the remote session. This method works regardless of the system running on the remote node.

When you terminate a remote session, the message "%REM-S-END, control returned to node _NODENAME::" is displayed, and you are returned to the system from which you made the remote node connection.

If the DECnet network has made intermediate connections for you and one of the intermediate systems goes down, DECnet either attempts to reroute the connection or waits a few seconds to determine whether the system will recover. If DECnet is able to recover the connection, the interruption may be so brief that you do not notice it, or it may last as long as 60 seconds. If DECnet cannot recover the connection, the remote session is terminated and the message "Path lost to partner" may be displayed.

See the Reference Section for more information about the DCL command SET HOST.

1.1.1.3 Dialing In

Dialing in allows you to communicate with your system by telephone. To dial in to your system, you need the following items:

- Modem (or data set)—A modem is a piece of hardware that is independent of the VMS system. The user's manual that comes with the modem should describe how to connect the modem to a telephone line and a terminal.
- Terminal—You cannot dial in unless the transmission speed (baud rate) of the terminal agrees with the baud rate of the modem and the modem terminal characteristic is set. To set the baud rate for VT200- and VT300-series terminals, select the "Comm" category in the Set-Up Directory. To set the baud rate for a VT100-series terminal, you must be in SET-UP B. To enter the Set-Up Directory, press the SET-UP key. Press the key labeled A/B to toggle to SET-UP B.

  To set the modem terminal characteristic, use the DCL command SET TERMINAL/MODEM. If your terminal has the modem terminal characteristic set (the DCL command SHOW TERMINAL lists the terminal characteristics set for your terminal), typing the SET TERMINAL/NOMODEM command causes the system to log you out.

  See your terminal's installation manual for information on using the Set-Up Directory to set the baud rate and other terminal characteristics.
- Manual login account—If your account is set up for automatic login, you cannot dial in to it. Either change the account to a manual login account (one where you must type your user name) or use a different account.
- Telephone number for the system—To dial in, you must know the telephone number for your system. Get the telephone number from your system manager.

Once the terminal is receiving the signal through the telephone line, follow the conventions your site has instituted for remote login. When communication is established, your system should respond with the DCL dollar sign prompt.

If you are managing your own system, you can ensure system security by disallowing dialup accounts with null passwords.

1.1.2 Logging Out of the System

When you finish using the system, always log out. This prevents unauthorized users from accessing your account and the system at large. It is also a wise use of system resources; the resources you no longer need are freed for use by others.

To log out, enter the LOGOUT command, which can be abbreviated as LO. You see a display confirming that you are logged out of the system that looks similar to the following:

```
$ LOGOUT
HARRIS logged out at 31-DEC-1988  12:42:48.12
```

You can log out of the system only when you are at the DCL prompt. You cannot enter the LOGOUT command while you are compiling or executing a program, using an editor (such as EDT or EVE), or running a utility (such as MAIL). First you must exit the program, editor, or utility and receive the DCL prompt. It is possible to exit by entering CTRL/Y, but this is not always advisable. See Section 1.3.2.2 for more information about using CTRL/Y.

To find out how much time you spent at the terminal (elapsed time), how much computer time you used (charged CPU time), and other accounting information, include the /FULL qualifier to the LOGOUT command as follows:

```
$ LOGOUT/FULL
SIMPSON logged out at 31-DEC-1988  12:42:48.12

Accounting information:
 Buffered I/O count:        8005   Peak working set size:    212
 Direct I/O count:           504   Peak virtual size:        770
 Page faults:               1476   Mounted volumes:            0
 Charged CPU time:0 00:00:50.01  Elapsed time:0 02:27:43.06
```

If you are logging out from a dialup terminal, enter the LOGOUT command with the /HANGUP qualifier. This command causes the system to break the connection to the dialup line after you log out.

1.2 Using the DIGITAL Command Language

The DIGITAL Command Language (DCL) is the language you use to communicate with the VMS operating system. DCL commands let you do the following:

- Get information about the system
- Work with files
- Work with disks, magnetic tapes, and other devices
- Modify your work environment
- Develop and execute programs
- Provide security and ensure that resources are used efficiently

DCL commands are usually verbs that describe what you want the system to do. In response to the DCL dollar sign ($) prompt, you enter a command (in upper- or lowercase). The following example shows how to enter the DCL command SHOW TIME:

```
$ SHOW TIME RET
```

The system responds by displaying the current date and time and returns the DCL prompt to indicate it is ready to accept another command:

```
31-DEC-1988  15:41:43
$
```

You can use DCL in the following two modes:

- Interactive—In interactive mode, you enter commands from your terminal. One command has to finish executing before you can enter another.
- Batch—In batch mode, the system creates another *process* to execute commands on your behalf. *Batch* jobs and network processes use DCL in batch mode. A process is an environment created by the system that makes it possible for you to work with the system. A batch job is a command procedure or program that is submitted to the operating system for execution as a separate user process. After you submit the command procedure for batch execution, you can continue to use your terminal interactively. (See Chapter 3 for more information about processes and batch jobs.)

  When you type a command and press RETURN, it is read and translated by the DCL interpreter.

1.2.1 DCL Command HELP

You can obtain online documentation for any DCL command by invoking the HELP facility. To use the HELP facility in its simplest form, enter the DCL command HELP. HELP displays a list of topics and the Topic? prompt. If you want to see information on one of the topics, type the topic name after the prompt. The system displays information on that topic. (Command and topic names can be abbreviated.)

If the topic has subtopics, HELP lists the subtopics and displays the Subtopic? prompt. If you want information on one of the subtopics, type the name after the prompt. If you want information on another topic, press RETURN. You can ask for information on another topic when HELP displays the Topic? prompt. If you want to exit the HELP facility, press RETURN again to return to DCL level. At any time, press CTRL/Z to exit.

If you know the command you need information about, type HELP and the command name.

If you need help but do not know what command or system topic to specify, enter the command HELP with the word HINTS as a parameter. Each task name listed in the HINTS text is associated with a list of related command names and system information topics.

Following is a sample HELP display for the DCL command SHOW USERS:

```
$ HELP SHOW USERS

SHOW

  USERS

     Displays the terminal name, username, and process
     identification code (PID)  of either specific interactive
     users or all interactive users on the system.

      Format:

      SHOW USERS [username]

  Additional information available:

  Parameters Command_Qualifiers
  /OUTPUT
  Examples

SHOW USERS Subtopic?
```

You can also obtain help while you are using an interactive utility. *Utilities* are programs that are invoked with DCL commands. To get help while you are using an interactive utility, type HELP at the utility prompt (and press RETURN) just as you would at DCL level. See Section 1.4 for more information about VMS utilities.

1.2.2 The DCL Command Line

DCL, like any language, has its own vocabulary and usage rules. The vocabulary consists of commands, parameters, and qualifiers, which are strung together in a way that DCL can interpret. The way in which the parts of a command line are put together is referred to as the *command line syntax*. Following is the general format for the DCL command line. (Items in square brackets [ ] are optional and may not be required by a specific command.)

[$] [label:] command [/qualifier[=value]...] [parameter[/qualifier...]]

The DCL command line can contain the following components:

$	The dollar sign is the DCL prompt. When you work interactively with DCL, DCL displays the prompt when it is ready to accept a command. When you write a command procedure, you must type the dollar sign at the beginning of each line.
Label	Identifies a line in a command procedure. Use labels only within command procedures, which are described in Chapter 6.
Command	Specifies the name of the command.
Parameter	Specifies what the command acts upon. You must position parameters in a specified order within the command. The DCL command descriptions in the Reference Section describe what parameter values are allowed for each command and where they must be placed. Examples of parameter values include file specifications, queue names, and logical names.
Qualifier	Modifies the action taken by the command. Some qualifiers can modify parameters. Some can accept values. The DCL command descriptions in the Reference Section indicate whether a specific qualifier can accept a value and what kind of value is acceptable.
Value	Modifies a qualifier and is often preceded by an equal sign. A value can be a file specification, a character string, a number, or a DCL *keyword*. A keyword is a word reserved for use in certain specified syntax formats. You must use keywords exactly as listed in the description of the particular DCL command you want to specify. For example, SYSTEM, OWNER, GROUP, and WORLD are DCL keywords. A DCL keyword can also have a value.

Following is an example of a DCL command line:

```
$ PRINT/COPIES=5 LAUNDRY.LIS RET
```

In the previous example, the elements are as follows:

- *$* is the DCL prompt.
- *PRINT* is a command.
- */COPIES* is a qualifier that modifies the command.
- *5* is a value that modifies the qualifier.
- *LAUNDRY.LIS* is a parameter (in this case, the parameter is a file specification).

In some cases (such as DELETE/ENTRY or SHOW QUEUE), a command is coupled with a parameter or qualifier. In these cases, the command and parameter or qualifier are used as a pair and cannot be separated. If you specify additional parameters or qualifiers, they must follow the command pair.

The following example shows a command pair that contains a command (SHOW) and a parameter (QUEUE). The additional parameter LN03_PRINT, which specifies the queue name whose jobs you want displayed, is specified after the command pair.

```
$ SHOW QUEUE LNO3_PRINT
```

Observe the following rules when entering DCL commands:

- You can use any combination of uppercase and lowercase letters. The DCL interpreter translates lowercase letters to uppercase. Upper- and lowercase

characters in parameter and parameter qualifier values are equivalent unless enclosed in quotation marks.

- Separate the command name from the first parameter with at least one blank space. Separate each additional parameter from the previous parameter with at least one blank space. Begin each qualifier with a slash (/); the slash serves as a separator and need not be preceded by blank spaces or tabs.

- You may need more than one line on your terminal screen to type a command line. Continue the command line onto the next line by terminating it with a hyphen and pressing RETURN. The system responds to this combination of a hyphen and RETURN with an underscore (_) followed by the dollar sign prompt; you continue typing the command line after this prompt. (A single command line cannot exceed 256 characters.) A line beginning with an underscore means that the system is waiting for your response, as shown in the following example:

  ```
  $ COPY/LOG FORMAT.TXT,FIGURE.TXT,ART_WORK.TXT -
  _$ SAVE.TXT
  ```

  Note that you must include the appropriate spaces between command names, parameters, and so on. Pressing RETURN after the hyphen does not add a space.

- A command line can contain a maximum of 128 elements (for example, a file specification or qualifier). Each element in a command must not exceed 255 characters. The entire command must not exceed 1024 characters after all symbols and lexical functions are converted to their values. (You use *symbols*, described in Chapter 5, to pass information to the system in an abbreviated manner. A *lexical function*, described in Chapter 6, obtains information from the system, including information about system processes, batch and print queues, and user processes, and then substitutes the result of the operation for itself.)

- You can abbreviate any command name by typing only the first four characters. You can abbreviate a command name to fewer than four characters as long as the abbreviated name remains unique among all DCL command names.

  For example, the following commands are equivalent:

  ```
  $ PR/C=2 FORMAL_ART.TXT
  $ PRINT/COPIES=2 FORMAL_ART.TXT
  ```

  In interactive mode, you will work faster if you abbreviate. Do not abbreviate commands in command procedures because your command procedure will be difficult to read. Also, the abbreviations might not be valid if new DCL commands are added at a later date.

Other rules governing the format of commands apply mainly to their use in command procedures. See Chapter 6 for more information about using commands in command procedures.

1.2.3 Prompting and System Defaults

Some items must be entered on the command line. If you do not enter them, the system displays a prompt and asks you to supply the missing information. In the following example, the TYPE command expects a file specification. Because a file specification is a required parameter, if you do not enter one, the system requests it. A line beginning with an underscore (_) means the system is waiting for your response.

```
$ TYPE
_File:   WATER.TXT
```

When you are prompted for an optional parameter, press RETURN to omit it. At any prompt, you can enter one or more of the remaining parameters and any additional qualifiers.

If you press CTRL/Z after a command prompt, DCL ignores the command and redisplays the DCL prompt.

Some items need not be specified on the command line. These are called defaults. When DCL does something by *default*, it assumes that you want a command to use certain values or to take certain actions without your having to explicitly specify them. In general, the values and actions are those considered normal or expected by users.

For example, if you do not specify the number of copies as a qualifier for the PRINT command, DCL uses the default value of 1. Unless you specify otherwise, DCL assumes that you have chosen the default. You can override this default behavior and print multiple copies of a file by specifying the following:

```
$ PRINT/COPIES=4 MYFILE.TXT
```

DCL supplies default values in several areas, including command parameters and qualifiers. Parameter defaults are described in the following section; qualifier defaults are described in Section 1.2.5.2.

1.2.4 Entering Parameters

DCL supplies default values for some command parameters. The parameters accepted by a command as well as the specific command parameter defaults supplied by DCL are described in each command description in the Reference Section. The following rules apply when specifying parameters:

- Square brackets ([ ]) indicate optional items. In the following example, you do not have to enter a file specification:

  DIRECTORY [file-spec]

  Anything not enclosed in square brackets is required. In the following example, you must enter a device name:

  SHOW PRINTER device-name

- Place required parameters to the left of optional parameters.
- Precede an output file parameter with an input file parameter. In the following example, the input file, LISTS.TXT, is copied to the output file, FORMAT.TXT:

  ```
  $ COPY LISTS.TXT FORMAT.TXT
  ```

  The following example reverses the order of the parameters, copying the input file, FORMAT.TXT, to the output file, LISTS.TXT:

  ```
  $ COPY FORMAT.TXT LISTS.TXT
  ```

- A parameter can be one item or a series of items. If you enter a series of items, separate them with commas (,) or plus signs (+). Any number of spaces or tab characters can precede or follow a comma or a plus sign. Note that some commands regard the plus sign as a concatenator, not as a separator. The parameter section of each DCL command description in the Reference Section describes how each command interprets commas and plus signs.

  The following command syntax line shows that you can optionally enter a list of files as the parameter:

  DELETE file-spec[,...]

  The following example shows how to specify a list of parameters. Here, three files are copied to a fourth file. The three file specifications—PLUTO.TXT, SATURN.TXT, and EARTH.TXT—constitute the first parameter. PLANETS.TXT is the second parameter.

  ```
  $ COPY PLUTO.TXT,SATURN.TXT,EARTH.TXT PLANETS.TXT
  ```

1.2.5 Entering Qualifiers

The qualifiers accepted by a command are described in each command description in the Reference Section. The DCL command description also indicates whether a qualifier accepts a value and what kind of value is required.

You can abbreviate any qualifier name by typing only the first four characters (not counting the slash). You can use fewer than four characters to abbreviate a qualifier name as long as the abbreviated name remains unique among all qualifier names for the same command.

Although you are never required to specify a qualifier, commands have defaults automatically applied. You need to be aware of the defaults that apply for each qualifier. The following sections describe types of qualifiers and qualifier defaults.

1.2.5.1 Types of Qualifiers

The three types of qualifiers are as follows:

- Command qualifiers—A command qualifier modifies a command. Although it is a good practice to place the qualifier after the command name (or, if you are specifying multiple qualifiers, after other command qualifiers that follow the command name), a command qualifier can appear anywhere in the command line.

  In the following example, /QUEUE is a command qualifier. The files SATURN.TXT and EARTH.TXT are queued to the LN03_PRINT queue.

  ```
  $ PRINT/QUEUE=LN03_PRINT SATURN.TXT,EARTH.TXT
  ```

- Positional qualifiers—A positional qualifier can modify commands or parameters and has different meanings depending on where you place it in the command string. If you place a positional qualifier after the command but before the first parameter, it affects the entire command string. If you place a positional qualifier after a parameter, it affects only that parameter.

  In the following example, the first PRINT command requests two copies of the files SPRING.SUM and FALL.SUM. The second PRINT command requests two copies of the file SPRING.SUM, but only one copy of FALL.SUM.

  ```
  $ PRINT/COPIES=2 SPRING.SUM,FALL.SUM
  $ PRINT SPRING.SUM/COPIES=2,FALL.SUM
  ```

- Parameter qualifiers—A parameter qualifier can be used only with certain types of parameters, such as input and output files.

  For example, the BACKUP command accepts several parameter qualifiers that apply only to input and output file specifications. In the following example, the /CREATED and /BEFORE qualifiers, which can be specified only with input files, select specific input files for the backup operation. (For the purposes of this example, multiple copies of the file MYFILE.TXT exist. Only those versions that were created before December 31, 1988, are selected for the backup operation.)

  ```
  $ BACKUP MYFILE.TXT/CREATED/BEFORE=31-DEC-1988 NEWFILE.TXT
  ```

1.2.5.2 Qualifier Defaults

When you omit a specific qualifier from the command line, the system responds with default behavior. For example, when you delete a file with the DELETE command, the system by default does not request confirmation of each delete operation. However, by specifying the DELETE/CONFIRM command, you can override that default behavior and request that you be prompted for confirmation before each file is deleted.

You can specify qualifiers in several ways. The qualifier syntax required by a specific DCL command is given in the command descriptions in the Reference Section. The following paragraphs explain the syntax used to describe qualifiers and their defaults:

- Qualifiers with positive and negative forms—These qualifiers have a value of true or false. You do not specify a value but indicate a true value by simply naming the qualifier. Negate the qualifier by inserting the prefix NO.

  For example, the /CONFIRM qualifier can be expressed positively or negatively. If you omit the qualifier from the command line, the default action is /NOCONFIRM. The syntax for the /CONFIRM qualifier is given in a DCL command description as follows:

  /CONFIRM
  /NOCONFIRM (default)

- Qualifiers that require values—If you use a qualifier that accepts a value, you must specify a value. If you omit the qualifier completely, the default value is applied. For example, if you use the /COPIES qualifier, you must provide a numeric value. If you omit the /COPIES qualifier, the default is /COPIES=1. The syntax for the /COPIES qualifier is given in a DCL command description as follows:

  /COPIES=n

  If the qualifier accepts a list of values, you must enclose the values in parentheses and separate them with commas as follows:

  ```
  $ DELETE/ENTRY=(230,231) LN03_PRINT
  ```

  The command deletes jobs 230 and 231 from the queue LN03_PRINT.

- Qualifiers that accept value and positive/negative combinations—Some qualifiers combine value and positive/negative characteristics so that the qualifier both accepts a value and allows you to negate the qualifier by inserting the prefix NO. For example, the SET TERMINAL command permits the following choices for the /PARITY qualifier:

  ```
  $ SET TERMINAL/PARITY=EVEN
  $ SET TERMINAL/PARITY=ODD
  $ SET TERMINAL/NOPARITY
  ```

- Qualifiers that affect command execution only if specified—The qualifier has no corresponding default. For example, the /BY_OWNER qualifier does not affect the command if it is not specified. The syntax for the /BY_OWNER qualifier is given in a DCL command description as follows:

  /BY_OWNER

- Qualifiers that override other qualifiers—Sometimes a command has a qualifier that is automatically applied as a default. Other qualifiers are available to override the default qualifier.

  For example, the /BRIEF qualifier is applied by default when you specify the DIRECTORY command. That is, the DIRECTORY command generates a listing that includes only the file name, file type, and version number of each file in the directory. You must specify the /FULL qualifier to generate a listing that

includes the file name, file type, and version number as well as the number of blocks used, the date of the file's creation, the date the file was last backed up, and so on.

Some commands contain conflicting qualifiers that cannot be specified in the same command line. If you use incompatible qualifiers, the command interpreter usually displays an error message. The command descriptions in the Reference Section indicate which qualifiers cannot be used together.

1.2.6 Entering Dates and Times as Values

Certain commands and qualifiers accept date and time values. You can specify these values in one of the following formats:

- Absolute time
- Delta time
- Combination time (combines absolute and delta time formats)

The DCL command descriptions in the Reference Section indicate the time formats accepted by individual commands and qualifiers.

1.2.6.1 Absolute Time

Absolute time is a specific date or time of day. The format for an absolute time is as follows:

[dd-mmm-yyyy][:][hh:mm:ss.cc]

The fields are as follows:

Field	Meaning
dd	Day of the month; an integer in the range 1–31
mmm	Month; JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC
yyyy	Year; an integer
hh	Hour; an integer in the range 0–23
mm	Minute; an integer in the range 0–59
ss	Seconds; an integer in the range 0–59
cc	Hundredths of a second; an integer in the range 0–99

You can truncate the date or the time on the right. However, if you are specifying both date and time, you must include a colon between them. The date must contain at least one hyphen. You can omit any of the fields within the date and time as long as you include the punctuation marks that separate the fields. A truncated or omitted date field defaults to the corresponding fields for the current date. A truncated or omitted time field defaults to zero. If you specify a past time in a command that expects the current or a future time, the current time is used.

You can also specify an absolute time as one of the following keywords:

Keyword	Meaning
TODAY	The current day, month, and year at 00:00:00.0 o'clock
TOMORROW	00:00:00.00 o'clock tomorrow
YESTERDAY	00:00:00.00 o'clock yesterday

Some examples of absolute time specifications follow:

Time Specification	Result
31-DEC-1988:13	1 P.M. on December 31, 1988
31-DEC	Midnight at the beginning of December 31 this year
15:30	3:30 P.M. today
31-	Midnight on the 31st day of the current year and month
31-::30	12:30 A.M. on the 31st of this month

1.2.6.2 Delta Time

Delta time is an offset (a time interval) from the current date and time to a time in the future. The general format of a delta time is as follows:

[dddd-][hh:mm:ss.cc]

The fields are as follows:

Field	Meaning
dddd	Number of days; an integer in the range 0–9999
hh	Number of hours; an integer in the range 0–23
mm	Number of minutes; an integer in the range 0–59
ss	Number of seconds; an integer in the range 0–59
cc	Number of hundredths of seconds; an integer in the range 0–99

You can truncate a delta time on the right. If you specify the number of days, include a hyphen. You can omit fields within the time as long as you include the punctuation that separates the fields. If you omit the time field, the default is zero.

Some examples of delta time specifications follow:

Time Specification	Result
3-	3 days from now (72 hours)
3	3 hours from now
:30	30 minutes from now
3-:30	3 days and 30 minutes from now
15:30	15 hours and 30 minutes from now

1.2.6.3 Combination Time

To combine absolute and delta time, specify an absolute time plus (+) or minus (-) a delta time. The format for combination time is as follows:

"[absolute time][+delta time]"
or
[absolute time][-delta time]

The variable fields and default fields for absolute and delta time values are the same as those described in the preceding sections. The delta time value must always be preceded by a plus or minus sign. (Note that the minus sign is the same keyboard key as the hyphen.) Whenever a plus sign precedes the delta time value, enclose the entire time specification in quotation marks. Also, you can omit the absolute time value. If you do, the delta time is offset from the current date and time.

Some examples of combination time specifications follow:

Time Specification	Result
"+5"	5 hours from now
"+:5"	5 minutes from now
-:5	Current time minus 5 minutes
-1-00	Current time minus 1 day. The minus sign (-) indicates a negative offset. The dash (-) separates the day from the time field.

If a qualifier is described as a value that may be expressed as an absolute time, a delta time, or a combination of the two, you must specify a delta time as if it were part of a combination time. For example, to specify a delta time value of five minutes from the current time, use "+:5" (not "0-0:5").

1.3 Entering and Editing DCL Commands

At the DCL level, you can use many individual keys and key combinations to change what you type, to recall commands, or to display information. Table 1–1 lists the keys that allow you to enter and edit DCL commands. Sections that follow describe these keys in greater detail.

DCL also provides you with shortcuts that simplify the typing of commands and command lines. You can establish symbols to use in place of command names and entire command strings. You can define keys, which enable you to enter commands with fewer keystrokes. These shortcuts are described in Section 1.3.7 and Section 1.3.8.

Table 1–1: Keys That Execute Terminal Functions

Key	Function
Keys That Enter DCL Commands	
CTRL/Z and F10[1]	Signals the end of the file for data entered from the terminal. CTRL/Z is displayed as "Exit."
RETURN	Sends the current line to the system for processing. (On some terminals, the RETURN key is labeled CR.) Before a terminal session, RETURN initiates a login sequence.
Keys That Interrupt DCL Commands	
CTRL/C and F6[1]	During command entry, cancels command processing. CTRL/C is displayed as "Cancel."
CTRL/T	Momentarily interrupts terminal output to display a line of statistical information about the current process. This display includes your node and user name, the time, the name of the image you are running, and information about system resources used during your current terminal session. You can also use the CTRL/T key to determine whether the system is operating. CTRL/T does not return information if the system is temporarily unresponsive or if your terminal is set to NOBROADCAST. In order to use CTRL/T, SET CONTROL=T must be enabled either in the system login command procedure or by you, either interactively or in your login command procedure.
CTRL/Y	Interrupts command processing. CTRL/Y is displayed as "Interrupt." You can disable CTRL/Y with the command SET NOCONTROL=Y.

[1]This key is available only on an LK201 keyboard.

Table 1-1 (Cont.): Keys That Execute Terminal Functions

Key	Function
Keys That Interrupt DCL Commands	
	Under most conditions, CTRL/Y returns you to the DCL prompt. The program running is still active. You can enter any built-in command (described in Section 1.3.2) and then continue the program with the CONTINUE command. (Press CTRL/W to refresh the screen after you enter the CONTINUE command.)
Keys That Recall Commands	
CTRL/B and Up arrow	Recalls up to 20 previously entered commands.
Down arrow	Displays the next line in the recall buffer.
Keys That Control Cursor Position	
<X, DELETE	Deletes the last character entered at the terminal. (On some terminals, the DELETE key is labeled RUBOUT.) The DELETE key also works when line editing is disabled.
CTRL/A, F14[1]	Switches between overstrike and insert mode. The default mode (as set with the SET TERMINAL/LINE_EDITING command) is reset at the beginning of each line.
CTRL/D and Left arrow	Moves the cursor one character to the left.
CTRL/E	Moves the cursor to the end of the line.
CTRL/F and Right arrow	Moves the cursor one character to the right.
CTRL/H, BACKSPACE, and F12[1]	Moves the cursor to the beginning of the line.
CTRL/I and TAB	Moves the cursor to the next tab stop on the terminal. The system provides tab stops at every eighth character position on a line. Tab settings are hardware terminal characteristics that, in general, you can modify. The TAB key also works when line editing is disabled.
CTRL/J, LINEFEED, and F13[1]	Deletes the word to the left of the cursor.
CTRL/K	Advances the current line to the next vertical tab stop.
CTRL/L	Causes the cursor to go to the beginning of the next page. This use of this key is ignored when line editing is enabled.
CTRL/R	Repeats the current command line and leaves the cursor positioned where it was when you pressed CTRL/R.
CTRL/U	Cancels the current input line.

[1]This key is available only on an LK201 keyboard.

Table 1-1 (Cont.): Keys That Execute Terminal Functions

Key	Function
Keys That Control Cursor Position	
CTRL/V	Turns off some of the line editing function keys. For example, if you press CTRL/V followed by CTRL/D, a CTRL/D is generated instead of the cursor moving left one character. CTRL/D is a line terminator at DCL level. When combined with CTRL/V, characters that are not line terminators have no effect. Examples are CTRL/H and CTRL/J. However, certain control keys, such as CTRL/U, retain their line editing functions.
CTRL/X	Cancels the current line and deletes data in the type-ahead buffer.
F7, F8, F9, F11	Reserved by DIGITAL.
Keys That Control Screen Display	
CTRL/O	Alternately suspends and continues display of output to the terminal. CTRL/O is displayed as "Output off" and "Output on."
CTRL/S	Suspends terminal output until CTRL/Q is pressed.
CTRL/Q	Resumes terminal output suspended by CTRL/S.
HOLD SCREEN[1] and NO SCROLL[2]	Suspends terminal output until the key is pressed again.

[1]This key is available only on an LK201 keyboard.

[2]This key is available only on a VT100 keyboard.

1.3.1 Entering a DCL Command

The RETURN key is recognized as a command line terminator. Once you type a command at the DCL prompt, press RETURN to terminate the line and send it to the DCL interpreter for execution. CTRL/Z is also recognized as a command line terminator.

When you enter a command at the terminal or execute an image that results in an error, the system displays an error message. Also, the system sometimes generates messages when a command has completed successfully. For interactive users, messages are normally displayed on the terminal; for batch job users, messages are written to the batch job log file.

Most system error messages have the following format:

```
%FACILITY-L-IDENT, text
```

The fields are as follows:

- FACILITY is a mnemonic for the program issuing the message.
- L is the first letter of the severity code; the severity level can be S (Success), I (Information), W (Warning), E (Error), or F (Fatal or severe error).
- IDENT is an abbreviation of the text.
- Text is an explanation of the error.

Suppress any component of the error message with the SET MESSAGE command. (See the description of the DCL command SET MESSAGE in the Reference Section for the qualifiers you need to specify to suppress individual components of the error message.) A SET MESSAGE command remains in effect until you enter the SET MESSAGE command again or log out. The following command suppresses the abbreviation of the explanatory text of the message:

```
$ SET MESSAGE/NOIDENTIFICATION
```

1.3.2 Interrupting and Canceling a DCL Command

After you enter a DCL command, you can temporarily interrupt its execution, run other commands, and then return to executing the command that was interrupted. To interrupt the execution of a command, use CTRL/T, CTRL/Y, or CTRL/C. These keys perform in different ways depending on the type of DCL command currently executing.

A command image is a program that is called by the DCL interpreter. (For example, COPY is a command image.) A command image can be privileged or nonprivileged. The VMS operating system, the system manager, or you may install a command image as privileged. Privileged command images may vary from system to system. Your system manager can tell you which command images on your system are privileged.

1.3.2.1 Using CTRL/T

CTRL/T interrupts execution of the command, displays a line of statistical information about the current process (node name, process name, system time, currently running image, elapsed CPU time, page faults, direct and buffered I/O operations, and pages in physical memory), and resumes command execution. You can use CTRL/T to interrupt a built-in command, or a privileged or nonprivileged command image. The following example shows how pressing CTRL/T interrupts and then resumes the copy operation:

```
$ COPY [JONES.MEMOS]TODAY.LIS URGENT.LIS
CTRL/T
SATURN::JONES 16:54:17 COPY CPU=00:00:00.54 PF=241 IO=47 MEM=141
$
```

In order to use CTRL/T, SET CONTROL=T must be enabled either in the system login command procedure or by you. You can enable CTRL/T in your login command procedure, or you can enable it interactively by entering SET CONTROL=T at DCL level. (To enable CTRL/T for the current session, you must enable it interactively.) Section 1.1 describes your personal login command procedure; Section 6.3 shows a sample personal command procedure.

1.3.2.2 Using CTRL/Y

When you use CTRL/Y to interrupt a nonprivileged command image, the interrupted command is temporarily suspended and control returns to the DCL interpreter. You see the DCL prompt. To resume execution of the interrupted command, type CONTINUE. Only built-in commands can be entered after CTRL/Y and before CONTINUE without disturbing the interrupted command. Entering any other type of command effectively cancels the interrupted one.

If you are interrupting a privileged command image, you can press CTRL/Y and enter the built-in commands SPAWN, ATTACH, and CONTINUE only, followed by any other command. Entering a command after CTRL/Y other than SPAWN, ATTACH, and CONTINUE effectively cancels the interrupted one.

You can immediately terminate the privileged or nonprivileged command image that you interrupted with CTRL/Y by entering one of the following:

- The STOP or EXIT commands. STOP suppresses any cleanup activities such as the display of error messages. EXIT executes any cleanup procedures before terminating.
- A command that invokes another command image (that is, a nonbuilt-in command), which removes the interrupted command image from memory.

1.3.2.3 Using CTRL/C

CTRL/C works like CTRL/Y in many cases. CTRL/C interrupts the execution of a built-in command. Command images, however, can create different definitions for CTRL/C, in which case pressing CTRL/C does not necessarily interrupt the command and return you to DCL level. For example, the TYPE command (a command image) defines CTRL/C as "cancel." Pressing CTRL/C while typing a series of files to the terminal halts the display of the current file and begins the display of the next file in the series, but does not interrupt the command.

1.3.3 Redirecting the Output of Commands

Many commands allow you to specify the /OUTPUT qualifier to redirect output. The following example shows how the display produced by the DCL command DIRECTORY is redirected to a new text file named FULL.LIS in your default directory:

```
$ DIRECTORY/FULL/OUTPUT=FULL.LIS
```

1.3.4 Recalling Commands

At DCL level, you can recall previously typed command lines and avoid the inconvenience of retyping long command lines. The recall buffer holds up to 20 previously entered commands. Once a command is displayed, you can reexecute or edit it.

Each of the following lets you display the commands stored in the recall buffer:

- CTRL/B
- Up and down arrow keys
- RECALL command

Pressing CTRL/B once recalls the previous command line. Pressing CTRL/B again recalls the line before the previous line, and so on to the last saved command line.

Pressing the up and down arrow keys recalls the previous and successive command, respectively. Press the arrow keys repeatedly to move through the commands.

To examine up to 20 previously typed command lines, type RECALL/ALL. Following is a sample display generated by typing RECALL/ALL:

```
$ RECALL/ALL

 1 SET DEFAULT DISK2:[MARSHALL]
 2 EDIT ACCOUNTS.COM
 3 PURGE ACCOUNTS.COM
 4 DIRECTORY/FULL ACCOUNTS.COM
 5 COPY ACCOUNTS.COM [.ACCOUNTS]*
 6 SET DEFAULT [.ACCOUNTS]
```

Having reviewed the available commands, you can recall a particular command line by typing RECALL and the number of the desired command. The following example shows how to recall the fourth command line stored in the recall buffer:

```
$ RECALL 4
```

After you press RETURN, the fourth command in the list is displayed at the DCL prompt. (The RECALL command itself is not placed in the buffer.)

You can also follow RECALL with the first characters of the command line you want to display. RECALL scans the previous command lines (beginning with the most recent one) and returns the first command line that begins with the characters you typed. For example:

```
$ RECALL E
```

After you press RETURN, the following command line is displayed:

```
$ EDIT ACCOUNTS.COM
```

You can also perform command recall with CTRL/B and the up and down arrow keys. If you are running a utility or an application program that uses VMS screen management software, you can also use these keys to perform command recall. Line editing must be enabled. Some utilities that have this feature are MAIL, DEBUG, SHOW CLUSTER, the System Dump Analyzer (SDA), and the VAXTPU editor.

1.3.5 Editing a DCL Command

Your terminal has a set of keys that you can use to edit a DCL command line. Command-line editing is most useful for modifying long command lines. You can edit command lines that contain typographical errors or command lines that you have recalled and want to modify.

There are many types of terminals, each with its own operating characteristics. In general, they all have standard line editing keys. Line editing keys (keys that let you edit the DCL command line) allow you to control cursor position and are listed in Table 1–1.

For some of the line editing keys to work, the SET TERMINAL/LINE_EDITING command must be in effect. To see whether or not line editing is enabled, enter the SHOW TERMINAL command, which displays your terminal's current characteristics. Use the SET TERMINAL command to change any of these characteristics. See the Reference Section for a description of the DCL command SET TERMINAL.

For example, the following command line contains one mistake that can be corrected easily using the line editing keys:

```
$ DILETE SCHEDULE.TXT;3
```

If you are using an LK201 keyboard (VT200- and VT300-series terminals), press the F12 key. If you are using a VT100-series terminal, press the BACKSPACE key. Notice that the cursor moves to the beginning of the line. Press the right arrow key once to position the cursor over the "I" in "DILETE." Type the letter E and the typographical error is corrected.

The preceding example assumes that the SET TERMINAL/OVERSTRIKE attribute is in effect, as it is by default. The OVERSTRIKE attribute allows you to replace the incorrect character by typing the correct character over it.

To insert characters in the command line without simultaneously deleting others, change the OVERSTRIKE attribute to the INSERT attribute. While you are editing a command line, you can set the OVERSTRIKE or INSERT attributes temporarily by pressing F14 (or CTRL/A). Use the SET TERMINAL command to set either attribute for your current terminal session.

1.3.6 Controlling Screen Display

Your terminal has several keys that permit you to suspend and resume the display of output to the terminal screen. These keys—CTRL/O, CTRL/Q, and CTRL/S—are useful when a large file is scrolling on your screen and you want to stop the display temporarily.

To suspend output to your terminal, press CTRL/S. To resume the output suspended by CTRL/S, press CTRL/Q. To toggle between suspending and resuming the output, type CTRL/O, which is alternatively displayed as "Output off" and "Output on."

The VT200- and VT300-series terminals also have a HOLD SCREEN key that you can press to alternately hold and resume screen output. On VT100 terminals, the NO SCROLL key performs this same function.

1.3.7 Representing DCL Commands with Symbols

When you specify parameters, multiple qualifiers, and values, one DCL command line can make for much typing. You can simplify your interaction with DCL and save time by establishing *symbols* to use in place of command names and entire command strings you type frequently. A symbol is a name that represents a numeric, character, or logical value. When you use a symbol in a DCL command line, DCL uses the value you assign to the symbol. By defining a symbol as a command line, you can execute the command by typing only the symbol name.

The following example equates the symbol ME to the DCL command SHOW ENTRY:

```
$ ME == "SHOW ENTRY"
```

After you equate a symbol to an expression (which can be a DCL command), the symbol assumes a new identity or value. In the previous example, the symbol ME assumes a new identity as the DCL command SHOW ENTRY. Once the two are equated, use the symbol ME in place of the SHOW ENTRY command as follows:

```
$ ME

   Jobname    Username   Entry   Blocks   Status
   -------    --------   -----   ------   ------
   STAFF      JONES        202       38   Printing
On printer queue SYS$PRINT
```

You can also equate long command strings to symbols. The following example equates the symbol LN03 with the command string shown:

```
$ LN03 == "PRINT/QUEUE=HUBBUB_LN03A/NOBURST/NOFEED/NOTIFY"
```

By defining a symbol interactively, you create a symbol that is in effect for the current session only. If you want that symbol to be in effect each time you log in, place the symbol definition in your login command procedure. See Chapter 5 for more information about defining symbols. See Section 1.1 and Section 6.3 for more information about creating a personal login command procedure.

1.3.8 Defining Terminal Keys

Key definitions let you customize your keyboard so you can enter DCL commands with fewer keystrokes. A *key definition* is a string of characters that you assign to a particular terminal key. When a key is defined, you can press it instead of typing the string of characters. A key definition usually contains all or part of a command line. When you press a defined key, the command is either displayed on your terminal or executed.

Some definable keys are automatically enabled for definition (like keys PF1 through PF4 and keys F17 through F20 on VT200- and VT300-series terminals). However, before you can define other keys, including KP0 (keypad 0) through KP9 and the keypad keys PERIOD, COMMA, MINUS, and ENTER, you must enable them for definition by entering either the SET TERMINAL/APPLICATION_KEYPAD or the SET TERMINAL/NONUMERIC command. For a complete list of definable keys and for more information on how to create key definitions, see the description of the DCL command DEFINE/KEY in the Reference Section.

The following example shows how to equate the PF1 key to the PRINT command and the PF2 key to the qualifier /QUEUE=SATURN_LN03:

```
$ DEFINE/KEY PF1 "PRINT"
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ DEFINE/KEY PF2 "/QUEUE=SATURN_LN03"
%DCL-I-DEFKEY, DEFAULT key PF2 has been defined
```

When you press the PF1 key and then the PF2 key, the words PRINT/QUEUE=SATURN_LN03 are echoed and entered as if you had typed them. You need only supply the parameter, which is the file name of the file you want to print. When defining a command line with two or more keys, remember to include all the necessary spaces required in the command line syntax.

The informational message following the key definition indicates the key state for which the key is defined. Key states are described in Section 1.3.8.1. You can suppress the informational message using the /NOLOG qualifier of the DEFINE /KEY command.

A key definition remains in effect until you redefine the key, enter the DELETE/KEY command for that key, or terminate the session. If you want to use a key definition each time you log in, place the key definition in your login command procedure. See Section 6.3 for more information about creating your personal login command procedure.

1.3.8.1 Key States

The same key can be assigned multiple definitions, as long as each definition is associated with a different state. A *key state* is a name you invent to remind you of the types of key definitions grouped under it. If you do not create any key states, all keys are defined in the DEFAULT state.

Specify the /SET_STATE qualifier to the DEFINE/KEY command to change the key state temporarily (the key state remains in effect until you press a definable key or terminate the command line). Use the /IF_STATE qualifier to the DEFINE/KEY to define a key for the specified state.

In the following example, the PF1 key in the DEFAULT state is defined to enter the PRINT command and to change the key state to PRINTERS. The MINUS key is defined in the PRINTERS state to enter the /QUEUE=LN03_PRINT qualifier. The COMMA is defined in the PRINTERS state to enter the /QUEUE=LINE_PRINT qualifier. (Remember to enter the DCL command SET TERMINAL/APPLICATION_KEYPAD to enable keypad key definitions.) The /TERMINATE qualifier places a carriage return after the text; when you press the key, the system attempts to execute the command line.

```
$ DEFINE/KEY/SET_STATE=PRINTERS PF1 "PRINT"
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ DEFINE/KEY/TERMINATE/IF_STATE=PRINTERS MINUS "/QUEUE=LN03_PRINT"
%DCL-I-DEFKEY, PRINTERS key MINUS has been defined
$ DEFINE/KEY/TERMINATE/IF_STATE=PRINTERS COMMA "/QUEUE=LINE_PRINT"
%DCL-I-DEFKEY, PRINTERS key COMMA has been defined
```

To change a key state permanently (until you log out or change the state again), specify the /LOCK and /SET_STATE qualifiers to the DEFINE/KEY command, or specify the /STATE qualifier to the SET KEY command. After permanently changing the key state, you can recall the DEFAULT key state. However, the system does not provide a mechanism that allows you to determine whether the DEFAULT state was the previous key state.

Because you cannot determine the previous key state after permanently changing the key state, you may want to use the following steps to extend the duration of a temporary state:

1. Use the /SET_STATE qualifier to the DEFINE/KEY command to change your key state temporarily.
2. Each time you define a key for that temporary state, use the /SET_STATE qualifier to reset the temporary state.

1.3.8.2 Examining and Deleting Keys

To examine the key definitions you have created, enter the SHOW KEY command. Specify the /DIRECTORY qualifier to display the states that you have defined as follows:

```
$ SHOW KEY/DIRECTORY
DEFAULT
GOLD
```

Specify the /ALL and /FULL qualifiers to list all the keys in the states specified by the /STATE qualifier. The following example shows that the PF1 key has been defined to enter the DIRECTORY command. The PF2 key has been defined to enter the SET DEFAULT command and change the key state from DEFAULT to DIRECTORIES.

```
$ SHOW KEY/ALL/FULL/STATE=DEFAULT
DEFAULT keypad definitions:
  PF1 = "DIRECTORY"  (echo,terminate,noerase,nolock)
  PF2 = "SET DEFAULT" (echo,noterminate,noerase,nolock,state=DIRECTORIES)
```

To delete a particular key definition, enter the DELETE/KEY command, as shown in the following example:

```
$ DELETE/KEY PF1
%DCL-I-DELKEY, DEFAULT key PF1 has been deleted
```

The following example shows how to delete all the keys defined in the GOLD state:

```
$ DELETE/KEY/ALL/STATE=GOLD
%DCL-I-DELKEY, GOLD key PF2 has been deleted
%DCL-I-DELKEY, GOLD key PF3 has been deleted
```

1.4 Utilities

A utility is a program that provides a service. Utilities are invoked with DCL commands. Some utilities—*interactive utilities*—provide a special environment from which you can perform a specific set of tasks. You work interactively with these utilities by entering subcommands and other information in response to the utility's prompt. For example, MAIL is an interactive utility; it has its own prompt and subcommands.

Other utilities are noninteractive. When you invoke a noninteractive utility, it occupies your terminal and executes a task. When the task is complete, you are returned to DCL level and your terminal is once again available. The SORT/MERGE and the LIBRARIAN utilities are two examples of noninteractive utilities.

Some utilities, both interactive and noninteractive, prompt you for a file name. When you are using such a utility (for example, BACKUP, MESSAGE, PATCH, and SORT /MERGE), you can add qualifiers to the DCL command line to tailor the utility to your specific needs, as shown in the following example:

```
$ BACKUP/RECORD/IMAGE/LOG [RET]
_From:
```

To exit from a utility and return to DCL level, type EXIT (and press RETURN) or press CTRL/Z in response to the utility prompt.

The following sections describe the interactive VMS Mail Utility, the VMS Phone Utility, and the Sort/Merge Utility.

1.4.1 Using the Mail Utility

The interactive VMS Mail Utility (MAIL) allows you to send messages to and receive messages from other users on your system or on any other VAX computer that is connected to your system by means of DECnet–VAX. You can also file, forward, delete, reply to, and print messages that you have received.

To invoke the Mail Utility, enter the DCL command MAIL at the DCL prompt. The MAIL prompt appears, signaling that the utility is ready to accept subcommands as follows:

```
$ MAIL
MAIL>
```

For more information about the MAIL commands and qualifiers, see the description of the Mail Utility in the Reference Section or type HELP at the MAIL prompt.

To exit from MAIL, enter the MAIL command EXIT or press CTRL/Z. Note, however, that if you are entering the text of a message, CTRL/Z sends the message. If you wish to cancel the send operation without exiting from MAIL, press CTRL/C.

1.4.1.1 Creating a Mail Subdirectory

When you receive mail messages, they are usually written to files named MAIL$xxxxxxxxxx.MAI located in your top level directory. To avoid the display of these MAI files in your top level directory, use the MAIL command SET MAIL_DIRECTORY, which creates a mail subdirectory and moves all your MAI files to that subdirectory. (The MAIL command SHOW MAIL_DIRECTORY displays the name of the subdirectory that contains all your MAI files.) To move the MAI files from a subdirectory back to your top level directory, use the SET NOMAIL_DIRECTORY command.

1.4.1.2 Sending Mail

You can create and send a mail message interactively to one user or many users with the Mail Utility. Also, you can send a file to other users from within MAIL or from DCL level.

Sending a Message

To send a mail message to any user on your system, invoke the Mail Utility and specify the MAIL command SEND. MAIL prompts you for the name of the user receiving the message, the subject of the message (optional), and the text of the message (optional). The following example sends a message to THOMPSON:

```
MAIL> SEND
To:     THOMPSON
Subj:   Meeting on January 9
Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
I have some new ideas for the Hubbub Cola account.  Let me know when
you're available to talk about them.

--Jeff
```

Press CTRL/Z to send the message. If you decide not to send the message, press CTRL/C. Doing so cancels the send operation without exiting from MAIL.

You can send the same message to several users. To do so, separate their user names with commas, as shown in the following example:

```
MAIL> SEND
To:     THOMPSON,JONES,BARNEY
Subj:   Meeting on January 9
```

If your computer system is part of a network, you can send mail to any other user on the network. If you are sending mail to someone not on your node, you must enter their node name and user name at the To: prompt. (See Section 1.1.1.2 for more information about nodes.) You can address the mail message to the intended recipient on the remote node using the following format:

nodename::username

The following example shows how to send a message to user HIGGINS on node CHEETA:

```
MAIL> SEND
To: CHEETA::HIGGINS
```

MAIL will notify you if the network connection to the remote node is not available.

You may want to use a VMS text editor to compose your message before you send it interactively. (A text editor allows you to enter text from the keyboard and use editing commands to modify that text. See Chapter 8 for a description of the EVE and EDT text editors.) To do so, specify the /EDIT qualifier with the SEND command as shown in the following example:

```
MAIL> SEND/EDIT
```

After you respond to the To: and Subj: prompts, MAIL invokes the text editor. By default, MAIL invokes the EDT editor. (Section 1.4.1.8 describes how to change the default editor.)

If you see an asterisk (*) after you enter the subject line and press RETURN, press the C key to enter the screen editor. To send the message, exit from the editor by pressing CTRL/Z and entering the EXIT command; to cancel the send operation, exit from the editor by pressing CTRL/Z and entering the QUIT command.

You can also use the /EDIT qualifier with the REPLY and FORWARD commands. By specifying /EDIT when you invoke MAIL, you can use the editor for send, reply, and forward operations during the ensuing mail session.

Sending a File

You can send a file to other users from within MAIL or from DCL level. The following example invokes MAIL and uses the MAIL command SEND to send a file:

```
$ MAIL
MAIL> SEND MEMO.TXT
To: EDGELL
Subj: Another memo
```

To send the file, press RETURN; to cancel the send operation, press CTRL/C or CTRL/Y. CTRL/C keeps you within the Mail Utility; CTRL/Y returns you to DCL level.

When you send a file from DCL level, MAIL is invoked, but you do not enter an interactive session, nor do you see the MAIL prompt. When the file is sent, you are automatically returned to DCL level. When you are sending a file in this way, the argument to the (optional) /SUBJECT qualifier must be enclosed in quotation marks if it contains any spaces or nonalphanumeric characters, as shown in the following example:

```
$ MAIL/SUBJECT="Another memo" MEMO.TXT CHEETA::EDGELL
```

To send the file, press RETURN; to cancel the send operation, press CTRL/C.

Sending a Message to a Distribution List

If you need to send one message to many users, you can create a file—called a *distribution list*—that contains a list of users. You then specify that file name rather than the individual user names when you send the message to those users. Use a text editor or the DCL command CREATE to create this file.

When you create a distribution list, type one user name per line. You can also include the names of other distribution lists by specifying an at sign (@) followed by the name of the distribution list. Exclamation points (!) delimit comments in programs and command procedures. DCL ignores everything to the right of the exclamation point when processing the line. For example:

```
! ALLBUDGET.DIS
!
! Budget Committee Members
@BUDGET   ! listed in BUDGET.DIS.
! Staff
HARRINGTON       ! me
BRUTUS::WILSON   ! Martha Wilson
PORTIA::RIPLEY   ! Roy Ripley
```

If the file BUDGET.DIS is not in the same directory as the new distribution list file you are creating, include the file specification for BUDGET.DIS in the new distribution file. (The file specification gives the system all the information necessary to locate a file. Depending on where you create ALLBUDGET.DIS, you may have to specify the device and directory in which BUDGET.DIS is located. See Chapter 2 for more information on file specifications.)

To send a message to a distribution list from within MAIL, type an at sign and the file name at the To: prompt. For example:

```
MAIL> SEND
To:      @ALLBUDGET
Subj:    Tomorrow's Meeting
```

By default, the system looks for a distribution list file with the file type DIS. If the file containing your distribution list has a different file type, you must specify the file name and file type at the To: prompt. If you invoke MAIL while in one directory and the file containing the distribution list is in another, enter the distribution list's file specification at the To: prompt.

1.4.1.3 Reading Mail

Invoke MAIL to read an old or new mail message. Messages you receive are stored in mail files, which have a default file type of MAI. Your default mail file, MAIL.MAI, is created in your top level directory the first time you receive a mail message.

By default, MAIL provides *folders*. New messages are automatically placed in a folder called NEWMAIL; old messages are held in a folder called MAIL. You can move between these folders to read old or new mail messages.

Reading New Messages

When you are logged in and receive a mail message, notice of the new message appears on your screen. (You can screen out notification of incoming messages by specifying the DCL commands SET TERMINAL/NOBROADCAST or SET BROADCAST=NOMAIL.) For example, a message sent by user FELLINI appears as follows:

```
New mail from FELLINI
```

If you are part of a DECnet-VAX network and someone on a remote node sends you mail, the sender's node and name are indicated.

If you have new mail, you are notified when you log in and when you invoke MAIL. To read a new message, invoke MAIL. MAIL displays the number of mail messages received and prompts for a command, as shown in the following example:

```
$ MAIL

You have 1 new message.

MAIL>
```

To read the new message, press RETURN. The message appears on your screen as follows:

```
#1                    31-DEC-1988  14:12:27          NEWMAIL

From:  FELLINI
To:    JONES
Subj:  Sales presentation on January 9

The meeting to discuss the Hubbub Cola account has been moved
from our conference room to the auditorium.  Dress to impress.

MAIL>
```

You may have another new message. To read your next new message, press RETURN at the MAIL prompt. Pressing RETURN in MAIL is equivalent to specifying the READ command without parameters. When you have read all your new messages, MAIL issues the message "%MAIL-E-NOMOREMSG, no more messages," and continues to prompt for commands until you exit by entering EXIT or pressing CTRL/Z.

If you receive a mail message while you are in MAIL, enter the READ/NEW command to read the new message.

Reading Old Messages

If you have just read a new message and want to reread an old message, enter the following:

```
MAIL> SELECT MAIL
```

This command selects the MAIL folder. The SELECT command allows you to move between folders. Once you are in the MAIL folder, press RETURN at the MAIL prompt or use the READ command to read the old message. The first message (numbered 1) in your default mail file appears on your screen. Press RETURN to display the next message. If the message is too long to display on one screen, press RETURN to display the next part of the message. To skip part of a message and display the next message, type NEXT, which can be abbreviated to "N."

You can display a list of all messages within the current mail folder by entering the DIRECTORY command. You can then display a particular message by entering the READ command and the number of the message, as shown in the following example:

```
MAIL> DIRECTORY
                                                    MAIL
# From              Date            Subject
1 DOLCE::FELLINI    31-DEC-1988     Sales presentation on January 9
2 DOODAH::JONES     31-DEC-1988     status

MAIL> READ 2
```

You can also omit the READ command and enter just the number of the message.

If you have many messages, you can locate a particular message by using the SEARCH command to find a specified string. To search for a string, specify that string as a parameter to the SEARCH command, as shown in the following example:

```
MAIL> SEARCH "appointment"
```

The SEARCH command selects and displays the first message in the current folder that contains the specified string.

To search for a new string, specify the string as a parameter to the SEARCH command. Each time you specify a new string, the SEARCH command starts the search at message number 1. To continue searching the folder for messages that contain the specified string, use the SEARCH command without specifying a parameter.

1.4.1.4 Creating a File from a Mail Message

To copy a mail message to a text file, enter the EXTRACT command while you are reading the message. When you exit from MAIL, the file is listed in your current directory (unless you specify another directory). The following example shows how to create a file named JANUARY_MEETINGS.TXT containing the text of message number 3:

```
MAIL> READ 3
.
.
.
MAIL> EXTRACT/NOHEADER JANUARY_MEETINGS.TXT
%MAIL-I-CREATED, DISK1:[JONES]JANUARY_MEETINGS.TXT;1 created
MAIL>
```

The mail header is composed of the From, To, and Subject lines. Specifying the /NOHEADER qualifier deletes the mail header and copies only the text of the message to the file. If the message has more than one header (as does, for example, a forwarded message), only the last header is deleted.

Use the /APPEND qualifier to the EXTRACT command to copy a message to the end of an existing file. Use the /ALL qualifier to copy all the files in the current folder to an existing file.

1.4.1.5 Deleting Mail

To delete a mail message, either enter the DELETE command while you are reading the message or enter the DELETE command followed by the number (or range of numbers) of the message you want to delete. The following example deletes messages 4, 5, 6, 11, 12, 14, 15, 16, and 17. You can use either the hyphen (-) or the colon (:) to define the range of messages to be deleted.

```
MAIL> DELETE 4-6,11,12,14:17
```

When you delete a message, the message is moved to a folder called WASTEBASKET. During your interactive MAIL session, you can recover any deleted message by moving the message out of the wastebasket folder. (See Section 1.4.1.6 for information on moving messages between folders.) Deleted messages collect in the WASTEBASKET folder until you exit from the current mail file (either by exiting from MAIL or by specifying a different mail file). Once you exit from the current mail file, WASTEBASKET is emptied and the folder itself is deleted. (See Section 1.4.1.6 for a discussion of mail files.)

1.4.1.6 Organizing Mail with Folders and Files

By default, each user account has one mail file (called MAIL.MAI). MAIL helps you organize your messages by providing the following folders as they are required:

- NEWMAIL—Contains all messages that have not been read. If you invoke MAIL when you have a new message, you are placed into the NEWMAIL folder. Once you leave the NEWMAIL folder (either by exiting MAIL or by changing to another folder), MAIL moves any messages that have been read but not deleted to your MAIL folder and deletes the NEWMAIL folder if it is empty.
- MAIL—Contains messages that have been read but not deleted. If you invoke MAIL and have no new messages, you are placed into the MAIL folder.
- WASTEBASKET—Contains messages that have been deleted. This folder and its contents are deleted when you exit MAIL or specify a different mail file.

You can extend this organizational scheme by creating your own folders. Each folder can contain any number of messages.

Like the default folders, the folders you create are normally stored in the mail file MAIL.MAI. You can also create your own mail files; each mail file can contain any number of folders. Although you can create any number of mail files, you usually organize your messages by creating folders rather than by creating mail files.

Creating and Modifying Folders

The following MAIL commands allow you to create and modify folders:

- FILE—Files the current message in the folder you specify. If the folder does not exist, you are asked whether you want to create it. After being filed, the message is automatically deleted from the current folder.

- COPY—Places a copy of the current message into the folder you specify. If the folder does not exist, you are asked whether you want to create it. The following commands copy all messages containing the word MEETING from the current folder to a folder named SCHEDULE. After the commands are executed, you have two copies of each message, one in the current folder and one in folder SCHEDULE. The first command selects and displays the first message containing the word "meeting":

```
MAIL> SEARCH MEETING

MAIL> COPY SCHEDULE
Folder SCHEDULE does not exist.
Do you want to create it (Y/N, default is N)?Y
%MAIL-I-NEWFOLDER, folder SCHEDULE created
```

  This command selects and displays the next message containing "meeting":

```
MAIL> SEARCH

MAIL> COPY SCHEDULE
MAIL> SEARCH
%MAIL-E-NOTFOUND, no messages containing 'MEETING' found
```

- MOVE—Synonymous with the FILE command.

Selecting Folders

The name of the current folder is displayed in the top right corner of the screen each time you enter a READ or DIRECTORY command. You can work only with messages that are in your current folder.

To display a list of the folders in your current mail file, enter the DIRECTORY /FOLDER command, as shown in the following example:

```
MAIL> DIRECTORY/FOLDER
Listing of folders in SYS$LOGIN:[JONES]MAIL.MAI;1
     Press CTRL/C to cancel listing
MAIL                                MEETING_MINUTES
MEMOS                               PROJECT_NOTES
STAFF
```

To select a new folder as your current folder, use one of the following commands:

- SELECT—Selects the specified folder as the current folder.
- DIRECTORY—Selects the specified folder as the current folder and lists the messages in the folder.
- READ—Selects the specified folder as the current folder and displays the specified message (by default, the first message in the folder).

Deleting Folders

To delete a mail folder, delete all the messages in the folder or move them to another folder. The following example deletes the MUSIC folder:

```
MAIL> SELECT MUSIC
%MAIL-I-SELECTED. 2 messages selected
MAIL> DELETE/ALL
```

Creating and Accessing Mail Files

To create a mail file, move a message into the file by entering the COPY, MOVE, or FILE command as you would to create a folder. When MAIL prompts you for the name of the folder, specify the name of the mail file after the name of the folder.

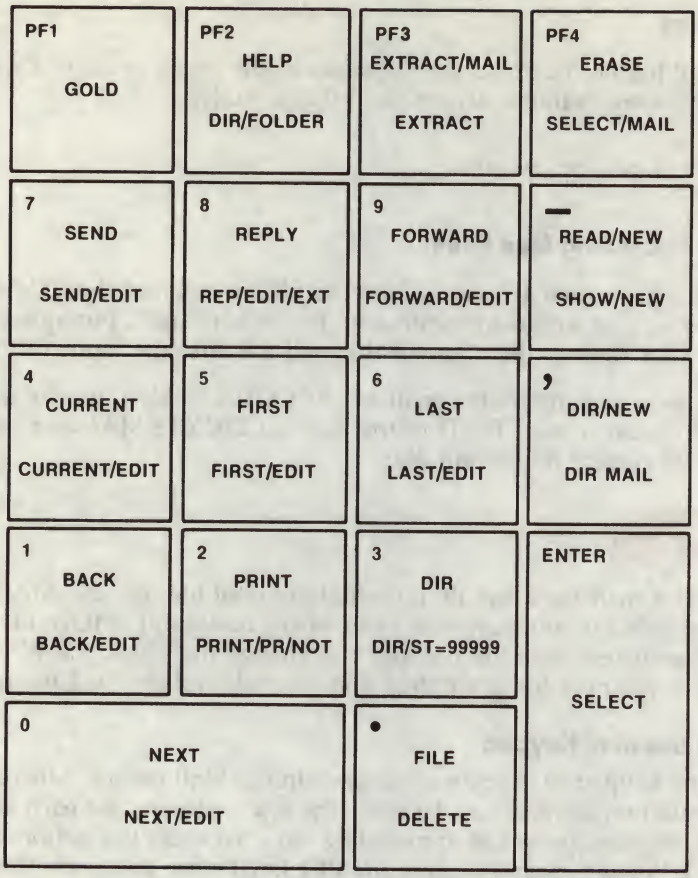
The following example creates the mail file ACCOUNTS.MAI, moves the current message into a folder named FEED in the file ACCOUNTS.MAI, and deletes the message from its current folder and file:

```
MAIL> MOVE
_Folder:  FEED [RET]
_File:  ACCOUNTS [RET]
```

To work within a mail file other than the default mail file, use the MAIL command SET FILE to specify the alternate file. (The MAIL command SHOW FILE displays the name of the current mail file.) When you change mail files, the WASTEBASKET folder of the current mail file is emptied and deleted, and the mail file is closed.

1.4.1.7 Using the Mail Keypad

You can use the keypad to execute commands in the Mail Utility. Most of the keypad keys can execute two commands. To enter the top command for each key shown in the following diagram, press the appropriate key. To enter the bottom command shown in the following diagram, press the PF1 key before you press the key.



ZK-1744-84

For example, to execute the MAIL command SEND, press the keypad key 7 (KP7). To execute the MAIL command SEND/EDIT, press the PF1 key first and then press KP7. (For more information on mail keypad commands, see the Mail Utility in the Reference Section.)

You can redefine the keypad keys to execute MAIL commands when you are in the Mail Utility. Defining keypad keys in MAIL is similar to defining keypad keys to execute DCL commands; see the DEFINE/KEY command in the Mail Utility in the Reference Section for more information.

1.4.1.8 Setting the Default Editor

By default, MAIL invokes the EDT editor when you specify the MAIL command SEND/EDIT. By entering the TPU parameter to the MAIL command SET EDITOR, you can specify that the TPU editor be invoked instead. (EVE is the default TPU editor.) The TPU editor remains your default MAIL editor (even if you log out of the system and log back in) until you enter the SET EDITOR EDT command.

The following example sets the default MAIL editor to TPU:

```
MAIL> SET EDITOR TPU
```

In the following example, the default MAIL editor has been set to TPU, and the MAIL command SEND/EDIT has been entered at the MAIL prompt. You see the following screen display:

```
Buffer   MAIN                                    | Insert | Forward
```

Enter the text of your message, using EVE commands to move around in the buffer, which is a temporary storage area that exists only during an editing session. Send the message by pressing CTRL/Z. (See Chapter 8 for information about using EVE. For a description of EVE commands, see the Reference Section.)

You can display the default MAIL editor by entering the MAIL command SHOW EDITOR, as shown in the following example:

```
MAIL> SHOW EDITOR
Your editor is TPU.
```

1.4.2 Using the Phone Utility

The VMS Phone Utility (PHONE) allows you to "talk" by way of your terminal screen to other users on your system or on any other VAX computer connected to your system by means of DECnet–VAX. The Phone Utility simulates the functions and features of a telephone. To invoke the Phone Utility, type PHONE at the DCL prompt. Your screen display splits horizontally into two sections. Your name is in the top section. At the switchhook character (the % sign), type the name of the person you want to call. Type the following to reach user SMITH on node CHEETA:

```
% CHEETA::SMITH
```

If you are calling another user on your node, or if your computer system is not part of a network, type only their user name.

PHONE rings the other party. If that person answers your call, their name appears in the bottom section. You can begin typing your conversation. If your call is not answered, you will be informed that the person is unavailable.

To answer a call from another user, invoke PHONE. Again, your terminal screen splits into two sections, with you in the top section. Enter the ANSWER command at the switchhook character. When you finish typing your conversation, enter the EXIT command or press CTRL/Z to exit from PHONE.

For more information about PHONE commands, type HELP at the PHONE prompt (%).

1.4.3 Using the Sort/Merge Utility

The VMS Sort Utility (SORT), invoked with the DCL command SORT, sorts records from one or more input files according to the fields you select and generates one reordered output file. The Sort Utility reorders records in a file (or files) so that they are in alphabetic or numeric order, either low to high (ascending) or high to low (descending), according to a portion of each record called the *key*. By default, the Sort Utility sorts on the first character of the first field in each record contained in the input file.

The VMS Merge Utility (MERGE), invoked with the DCL command MERGE, combines up to ten previously sorted files into one ordered output file. By default, MERGE does sequence checking to ensure the input files are in order. The sequence check stops the merge if a record is found to be out of order. To prevent sequence checking during the merge, specify the /NOCHECK_SEQUENCE qualifier.

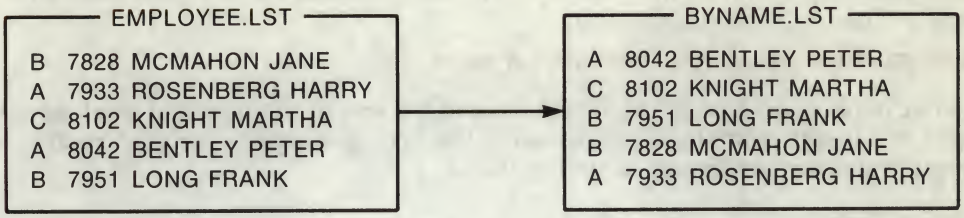
For more information about the SORT/MERGE parameters and qualifiers, see the Reference Section.

1.4.3.1 Sorting Records

A file record can be thought of as a line of text in a file. Record sorting, the default sort operation, keeps records intact and produces an output file consisting of complete records. Records can be subdivided into fields, which describe individual segments of the record. A field is specified by the starting position of its first character in the record and the length, in characters, of the field. You can sort records based on the contents of certain fields by specifying the field as a sort key.

The following example illustrates an ascending (the default) record sort based on that portion of each record starting at character position 8 and extending to the end of the record (the name):

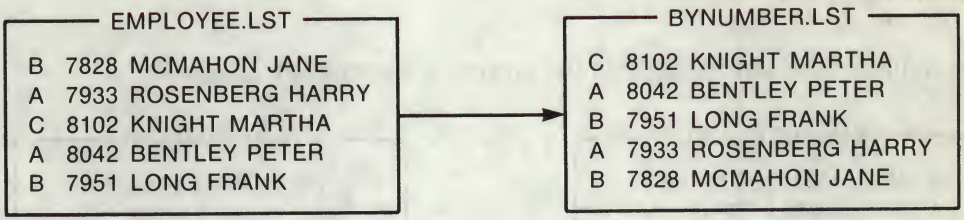
```
$ SORT/KEY=(POSITION=8,SIZE=15) EMPLOYEE.LST BYNAME.LST
```



ZK-1748-84

The following example sorts the same file in descending order using the field in character positions 3 through 6 (the number) as the sort key:

```
$ SORT/KEY=(POSITION=3,SIZE=4,DESCENDING) EMPLOYEE.LST BYNUMBER.LST
```



ZK-1749-84

The first parameter of the SORT command names the file or files to be sorted. Multiple files are treated as one large file for sorting purposes. The second parameter provides a name for the ordered output file that the sort will create. The following example sorts the records in two files, EMPLOYEE.LST and EMPLOYER.LST, and creates the ordered output file BYNAME.LST:

```
$ SORT EMPLOYEE.LST,EMPLOYER.LST BYNAME.LST
```

Single Key

By default, the SORT command assumes that a key field in a record has the following characteristics:

- Begins in the first position of a record
- Includes the entire record
- Contains character data
- Will be sorted in ascending order

Use the /KEY qualifier to specify characteristics of the key field other than those assumed by default.

In the following example, the /KEY qualifier specifies that the key field starts in position 8 and is 15 characters long:

```
$ SORT/KEY=(POSITION=8,SIZE=15) EMPLOYEE.LST BYNAME.LST
```

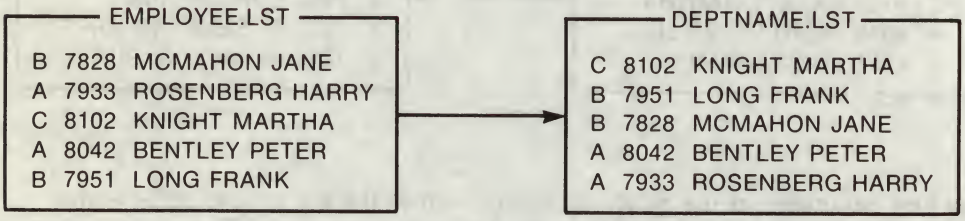
(If an actual key would have to extend beyond the end of the record to meet the size specification—for example, if the key is the last item in a variable-length format—the missing characters are treated as null characters.)

Multiple Keys

You can specify more than one key field, up to a limit of 255 characters. Each key can be ascending or descending. Specify multiple keys in the order of their priority in the sort. For example, the following command sorts records first on the value of position 1 in descending order, then on the value of positions 8 through 27 (or the end of the record) in ascending order:

```
$ SORT/KEY=(POSITION=1,SIZE=1,DESCENDING) -
_$ /KEY=(POSITION=8,SIZE=15) -
_$ EMPLOYEE.LST DEPTNAME.LST
```

The results of the sort specified in the preceding example are as follows:



ZK-1764-84

By default, records with identical keys are kept but not sorted predictably. To retain identical keys and arrange them according to the input file order, specify the /STABLE qualifier. To eliminate duplicate keys, specify the /NODUPLICATES qualifier.

1.4.3.2 Character Data Files

The SORT command assumes by default that the files to be sorted contain character data. Characters are sorted according to a collating sequence, which describes the order in which characters are arranged (A, B, C, and so on).

ASCII is the default collating sequence for character data. In general, ASCII orders numbers (0 through 9) first, then uppercase letters (A through Z), and then lowercase letters (a through z).

You can specify the EBCDIC collating sequence to generate an output file that is ordered in EBCDIC sequence (although it remains in ASCII representation). To use the EBCDIC collating sequence, specify the /COLLATING_SEQUENCE=EBCDIC qualifier.

The multinational collating sequence collates characters according to the international character set defined by DIGITAL (see Appendix A). The multinational collating sequence compares for different characters first, then for different diacritical forms of the same character (formed by using diacritical marks as part of "compose sequences" on VT200-series terminals), and then for different cases (uppercase or lowercase) of the same character. To use the multinational collating sequence, specify the /COLLATING_SEQUENCE=MULTINATIONAL qualifier.

NOTE: Use caution when using the multinational collating sequence to sort or merge files for further processing. Sequence-checking procedures in most programming languages compare numeric characters. Because the multinational sequence is based on actual graphic characters (and not the codes representing those characters), normal sequence checking will not work.

1.4.3.3 Noncharacter Data Files

If you sort files containing items other than character data, you must specify the data type of each key. Also, you must take care in calculating starting positions and sizes, because the items being compared may occupy more than one byte. For example, if you are sorting a file that contains 20 characters followed by 3 floating-point numbers in F_floating format, and the key is the last floating-point number, you must make the following specification:

```
$ SORT/KEY=(POSITION=29,F_FLOATING) STATS.RAW STATS.SOR
```

In the example, the character data occupies positions 1 through 20 (20 characters), the first F_floating-point number occupies position 21 through 24, the second F_floating-point number occupies positions 25 through 28, and the third F_floating-point number occupies positions 29 through 32. The size of the floating-point number is not specified (since it is fixed at 4 bytes).

1.4.3.4 Terminal Input

The records to be sorted or merged need not be in a file. You can enter the records directly from the terminal as you enter the SORT or MERGE command.

To enter the input records for a sort or merge operation from your terminal, specify SYS$INPUT as the input file parameter, qualifying it with the size of the longest record (in bytes) and the approximate size of the input file (in blocks). After you enter the command, enter the input records on successive terminal lines. Terminate each record by pressing RETURN. Terminate the file by pressing CTRL/Z.

The following example demonstrates a sort operation in which the input records to be sorted are entered directly from the terminal:

```
$ SORT/KEY=(POSITION=8,SIZE=15) -
_$ SYS$INPUT/FORMAT=(RECORD_SIZE=22,FILE_SIZE=10) BYNAME.LST
B 7828 MCMAHON JANE [RET]
A 7933 ROSENBERG HARRY [RET]
C 8102 KNIGHT MARTHA [RET]
A 8042 BENTLEY PETER [RET]
B 7951 LONG FRANK [RET]
[CTRL/Z]
```

1.4.3.5 Batch Job Submission

If you are sorting large files, you should consider submitting the sort operation as a batch job, since the sort will require some time. Batch jobs are programs or DCL command procedures that run independently of your current session. See Sections 3.1.2 and 3.1.4 for more information about command procedures and batch jobs, respectively.

If the records to be sorted are in a file, the command procedure you submit as a batch job must contain the SORT command and explicitly set your default directory or include the directory in the command file specifications. The following example submits the DCL command procedure SORTJOB.COM as a batch job. The text of the command procedure is shown following the command line:

```
$ SUBMIT SORTJOB

! SORTJOB.COM
!
$ SET DEFAULT [USER.PER]   ! Set default to location of input files
$ SORT/KEY=(POSITION=8,SIZE=15) EMPLOYEE.LST BYNAME.LST
```

You can include the input records in the batch job by placing them after the SORT command, one record per line, as shown in the following example. As with terminal input of records, you specify the input file parameter as SYS$INPUT and qualify it with the record size (in bytes) and the approximate file size (in blocks):

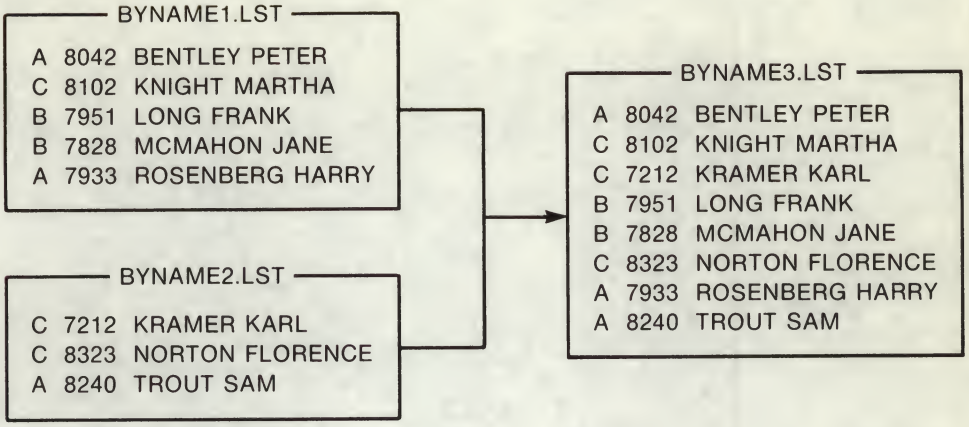
```
$ SUBMIT SORTJOB

! SORTJOB.COM
!
$ SET DEFAULT [USER.PER]
$ SORT/KEY=(POSITION=8,SIZE=15)-
SYS$INPUT-
/FORMAT=(RECORD_SIZE=22,FILE_SIZE=10)-
BYNAME.LST
B 7828 MCMAHON JANE
A 7933 ROSENBERG HARRY
C 8102 KNIGHT MARTHA
A 8042 BENTLEY PETER
B 7951 LONG FRANK
```

1.4.3.6 Merging Files

The MERGE command combines up to 10 sorted files into one ordered output file. The input files must all have the same format, and all must have been sorted on the same key fields.

The following example demonstrates the merging of two files based on the field in each record starting at position 8 and extending to the end of the record (the name field):

```
$ MERGE/KEY=(POSITION=8,SIZE=15) BYNAME1.LST,BYNAME2.LST BYNAME3.LST
```



ZK-1771-84

By default, MERGE does sequence checking to ensure the input files are in order. The sequence check stops the merge and reports an error if a record is found to be out of order. To prevent sequence checking during the merge, specify the /NOCHECK_SEQUENCE qualifier.

Chapter 2

Working with Files and Directories

In the VMS operating system, information is hierarchically stored. At the top of this hierarchy is the *master file directory* (MFD). Your *user file directory* (UFD) is listed in this master file directory, along with the user file directories of other users. Your user file directory (usually called username.DIR) is a file that points to your *top level directory*, which is also called your *login directory* or *default directory* because the system places you there by default when you log in. This top level directory contains the files and subdirectories that you have created or that have been created for you. It is from your top level directory that you perform most of your daily online tasks.

An MFD and UFDs are stored on physical devices called *disks*. The access path to a *file* is through the node and device, through a top level directory, through any *subdirectories*, and then to the file.

Your directory structure resembles a family tree. At the top is your top level directory, which branches off to files and to subdirectories, which branch still further. You can ascend and descend the directory structure to access your files and subdirectories. You can also access other directory structures that have been set up to allow public access. With the correct *process privileges*, you can also access files and directories on remote systems. Process privileges control what commands and functions you are authorized to execute from your account. See the *VMS System Manager's Manual* for more information about process privileges.

2.1 Files

A file contains information. This information can be machine-readable data that the computer understands. It can also be text you enter and manipulate. The text in the file might be the text of a document; a program that you can execute, written in a language such as C or Pascal; or a list of addresses. You can examine the data in these files by displaying the files on a terminal screen and printing them on paper.

Every file must have a file name or file type to identify it to both the system and you. A file also has a version number. This file information is specified using the following format:

filename.type;version

Taken together, these elements form a *file specification*. The following section describes the elements of a file specification and the rules for specifying these elements.

2.1.1 File Names, Types, and Versions

When you create a file, give it a name that is meaningful to you. The file name can be from 0 through 39 characters chosen from the letters A through Z (upper- or lowercase), the numbers 0 through 9, an underscore (_), a hyphen (-), or a dollar sign ($). Do not use a hyphen as the first or last character in the file name. Do not begin a file name with a dollar sign, although it is a legal character within the file name.

A file type identifies the nature of a file. The file type can be from 0 through 39 characters and must be preceded by a period. The rules for creating file names also apply to file types.

Including a file type is optional. With certain commands, if you omit the file type, the system applies a default value. Table 2–1 lists some of the more common default file types used by DCL commands. It also lists the default file types for some high-level language source programs.

Table 2–1: Default File Types

File Type	Contents
Default File Types for DCL Commands	
CLD	Command description file
COM	Command procedure file
DAT	Data file
DIS	Distribution list file for the MAIL command
DIR	Directory file
EDT	Startup command file for the EDT editor
EXE	Executable program image file created by the linker
HLP	Input source file for help libraries
JOU	Journal file created by the EDT editor
LIS	Listing file created by a language compiler or assembler; default input file for the PRINT and TYPE commands
LOG	Batch job output file
MAI	MAIL message file
MEM	Output file created by DIGITAL Standard Runoff (DSR)
OBJ	Object file created by a language compiler or assembler

Table 2-1 (Cont.): Default File Types

File Type	Contents
Default File Types for DCL Commands	
RNO	Input source file for DIGITAL Standard Runoff
SIXEL	Sixel graphic file
SYS	System image
TJL	Journal file created by the VAXTPU and ACL editors
TMP	Temporary file
TPU	Command file for the VAXTPU editor
TXT	Input file for text libraries or MAIL command output
Default File Types for Language Source Programs	
ADA	Input source file for the VAX Ada compiler
BAS	Input source file for the VAX BASIC compiler
B32	Input source file for the VAX BLISS-32 compiler
C	Input source file for the VAX C compiler
COB	Input source file for the VAX COBOL compiler
FOR	Input source file for the VAX FORTRAN compiler
MAR	Input source file for the VAX MACRO compiler
PAS	Input source file for the VAX Pascal compiler
PLI	Input source file for the VAX PL/I compiler

In addition to a file name and type, every file has a version number. Version numbers are decimal numbers from 1 to 32,767 that differentiate versions of a file. When you initially create a file, the system assigns it a version number of 1.

You may have several versions of a file. Unless you specify a version number, the system uses the highest existing version number of that file. When you modify that file, the system saves the original file and produces a modified output file. By default, this output file has the same name and type as the original, but the version number is incremented by one.

Version numbers must be preceded with a semicolon or a period. When the system displays file specifications, it generally displays a semicolon in front of the file version number.

The following example shows how to display the latest version of the file STAFF_VACATIONS.TXT. Because the system displays the latest version of a file by default, you can omit the version number from the file specification.

```
$ TYPE STAFF_VACATIONS.TXT
```

You can refer to versions of a file in a relative manner by specifying a zero or a negative version number. Specifying zero locates the latest (highest numbered) version of the file. Specifying -1 locates the next-most-recent version, -2 the version before that, and so on.

You can control the number of versions of a file by specifying the /VERSION_LIMIT qualifier to the DCL commands CREATE/DIRECTORY, SET DIRECTORY, and SET FILE.

2.1.2 File Characteristics

A file consists of records, each of which consists of a number of bytes of data. (Bytes are commonly used to represent characters.) A file's characteristics describe the physical layout of a file and determine how the file is treated during file operations. Specifically, file characteristics describe the following features of a file:

- File organization—Sequential, indexed, or relative.

  The records of a *sequential file* are arranged one after another in the order of creation. Records must be read from the file in order. The file must be rewritten (that is, another file or version of the file must be created) to update it.

  The records of an *indexed file* are arranged randomly and accessed through one or more indexes. An index contains a portion of each record called a key; the keys are arranged in sequence from lowest to highest (by binary, numeric, or ASCII value depending on data type); one key is called the primary key. You can read a record directly (randomly) by specifying an index and the value of one of its keys. You can read records sequentially by specifying an index—records are read in ascending sequence according to the key values for that index, starting with the current record. Update an indexed file in place by adding, deleting, or changing records. Indexed files require more space since, in addition to the data, the indexes must be stored.

  The records of a *relative file* are arranged in fixed-length, numbered cells. The cell numbers are used to determine the position of the record in the file. As with indexed files, you can read records sequentially or randomly. Typically, relative files are created and accessed by programs, rather than from DCL command level.

- Record format—Indicates the way all records in a file appear physically on the recording surface of the storage medium. Record format is defined in terms of record length and can be fixed length, variable length, variable length with fixed control area (VFC), or stream. All records in a fixed-length file are the same size. Records in a variable-length file vary in size. Records in a VFC file have a fixed-length header followed by a variable part. Note that VFC record format is not applicable for indexed files. Records with stream format are delimited with special control characters.

- Data type—Strictly speaking, a file does not have a data type, because programs processing a file must know how each item in the file is to be interpreted. However, a file whose records contain all character data (each item is one byte, interpreted according to ASCII conventions) is called a text, or character, file. A file whose data is formatted as integers, floating-point numbers, object code, or other non-ASCII data is called a binary file.

- Carriage control—New line (also known as "implied," "carriage return," or "CRLF"), FORTRAN carriage control, none, or print. New line places a carriage return and line feed at the end of each record when it is displayed or printed. FORTRAN carriage control uses the first character of each record to specify carriage-control information. "None" does not place carriage-control characters into a file; if you want to include control characters in the file, you must specify them as part of the data in the file. Note that the PRINT and TYPE commands interpret carriage-return, line-feed, and form-feed characters embedded in records. Print carriage control interprets the two bytes of each VFC record as prefix and postfix carriage-control information.

  Files you create using the editor or the CREATE command use new-line carriage control. Each time you press RETURN, you create a new record. When the file is printed or typed, each record appears on a new line. Files you create using the OPEN, WRITE, and CLOSE commands use print carriage control. Each WRITE command adds a new record (in VFC format) to the file.

- File size—The size of a sequential file with fixed-length records can be calculated by multiplying the number of records and the size of each record. Variable-length records require two extra bytes per record, and indexed files require space for the indexes. In addition to the files themselves, the VMS system uses disk space to store directory entries, file headers, and other file-maintenance information.

At DCL level, you normally deal with sequential, variable-length text files, although some commands permit access to indexed files. You can examine a file's characteristics with the /FULL qualifier of the DIRECTORY command, as shown in the following example:

```
$ DIRECTORY/FULL RECEIPTS.DAT

Directory DISK1:[JONES.TAXES]

RECEIPTS.DAT;15                 File ID: (103,75,0)
Size:     64/66              Owner:   [200,200]
Created:  02-JUN-1988 17:47:26.30
Revised: 31-DEC-1988 11:28:51.35 (2)
Expires:  <None specified>
Backup:  30-DEC-1988  22:48:08:23
File organization:   Sequential
File attributes:     Allocation=153, Extend=0 Global Buffer Count = 0
                     No version limit
Record format:       Variable length, maximum 82 bytes
Record attributes:   Carriage return carriage control
Journaling enabled:  None
File protection:     System:RWED, Owner:RWED, Group:RW, World:
Access Control List: None

Total of 1 file, 64/66 blocks.
```

The file size of the preceding example indicates that 64 blocks have been used out of the 66 allocated. (File size is the number of actual blocks used of the blocks that have been allocated; more will be allocated by the system as needed.) If you are only interested in the size of the file (or several files), use the /SIZE qualifier. The following example lists the number of blocks used by the files in one directory.

```
$ DIRECTORY/SIZE

Directory DISK1:[JONES.TAXES]

BILLING.DAT;31          62
LEGAL.TXT;9             20
LOCAL.DIS;2              4
PROPERTY.DIR;1           7
RECEIPTS.DAT;15         64
SALES.DIR;1              5

Total of 6 files, 162 blocks.
```

2.2 Directories

A directory is a special kind of file that catalogs (by name and location) a set of files. A directory file contains the following information for every file cataloged within it:

- The file name, type, and version number
- A pointer to the file header, which describes, among other things, the file's owner, protection code, and location

A directory file has the following format:

directory.DIR;1

For example, DOG.DIR;1 is a directory file. Because you cannot edit a directory file, all directory files have a version number of 1.

In addition to the file name, a *file specification* can include the directory in which the file is located. The following example shows the file specification used to display the file STAFF_VACATIONS.TXT located in the directory [JONES]:

```
$ TYPE [JONES]STAFF_VACATIONS.TXT
```

If you omit the directory name from the file specification, the current directory is assumed by default.

2.2.1 Directory Structure

Each disk contains a main directory that is set up by the system manager. This main directory is called the master file directory (MFD). The MFD contains a list of user file directories (UFDs). User file directories are files in the master file directory that point to top level directories. Your top level directory is also called your *login* or *default directory*. Unless your account has been specially modified to do otherwise, by default the system places you in your top level directory when you log in.

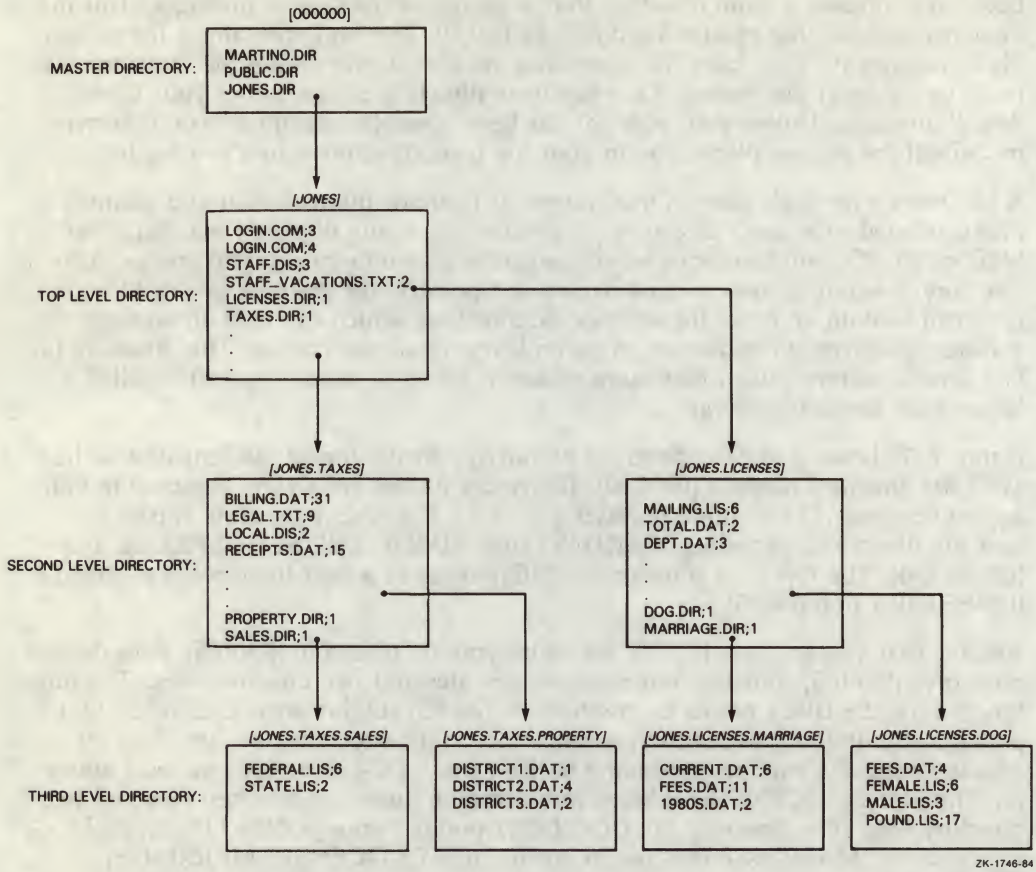
A UFD exists for each user on the system. It contains the names of and pointers to files cataloged in a user's directory. A *subdirectory* is any directory file that is not an MFD or a UFD. Subdirectories let you organize files into meaningful groups. Like a directory, a subdirectory contains names and pointers for the files cataloged within it. It can contain an entry for another subdirectory, which can contain an entry for another subdirectory, and so on to seven levels of subdirectories. This structure (a first level directory plus a maximum of seven levels of subdirectories) is called a *hierarchical directory structure*.

Figure 2–1 shows a sample directory hierarchy. At the top of the structure is the MFD. Its directory name is [000000]. (Directory names are always enclosed in either square brackets ([]) or angle brackets (<>).) Figure 2–1 contains entries for user file directories including MARTINO.DIR, PUBLIC.DIR, SCHULTZ.DIR, and JONES.DIR. The top level directory [JONES] exists as a user file directory named JONES.DIR;1 in [000000].

Assume that you are user JONES. At login, you are placed in [JONES], your default directory. [JONES] contains four nondirectory files and two directory files. The directory file TAXES.DIR;1 points to the [JONES.TAXES] subdirectory; LICENSES.DIR;1 points to the [JONES.LICENSES] subdirectory. (Subdirectories are specified by concatenating the subdirectory name to the name of the directory one level above it.) The [JONES.LICENSES] subdirectory contains three nondirectory files and two directory files. The directory file DOG.DIR;1 points to the [JONES.LICENSES.DOG] subdirectory; MARRIAGE.DIR points to the [JONES.LICENSES.MARRIAGE] subdirectory.

This sample directory structure is the basis for the examples in this chapter, which demonstrate how to ascend and descend the directory structure and how to access files within this structure.

Figure 2-1: Directory Structure



2.2.2 Directory Names

Use a named directory specification to refer to a directory. A named directory specification consists of a top level directory name that can be followed by a maximum of seven subdirectory names.

A named directory specification has the following format:

[directory.subdirectory[.subdirectory...]]

A directory name can contain up to 39 alphanumeric characters. Any characters valid for file names are also valid for directory names. Enclose the directory name in either square brackets ([]) or angle brackets (<>).

Default and *wildcard characters* can be applied. You use wildcard characters to apply DCL commands to multiple files rather than to one file at a time and to move around the directory structure. See Section 2.6.6.3 for more information about using wildcard characters in a named directory specification.

2.3 Devices

Files are stored on devices. In the VMS operating system, devices are classified as follows:

- Mass storage devices save the contents of files on a magnetic medium. Files saved this way can be accessed, updated, modified, or reused at any time. Disks and magnetic tapes are mass storage devices.
- Record-oriented devices read and write only single physical units of data at a time and do not provide online storage of the data. Terminals, printers, mailboxes, and card readers are record-oriented devices. (Printers and card readers are also called unit-record devices.)

A device name has the following three parts:

- The device type, which identifies the hardware device. (For example, an RP06 disk has the device type DB, and a TE16 magnetic tape has the device type MT.)
- A controller designator, which identifies the hardware controller to which the device is attached.
- The unit number, which uniquely identifies a device on a particular controller.

The files you commonly access are stored on disks or magnetic tape. Your user file directory (UFD) and your default directory with all your files and subdirectories are located on a disk. You can use a file specification that contains directory information only if the file is located on a disk. Magnetic tapes do not have directory structures. To obtain a file stored on tape, use a file specification that contains only file information.

If you want to access a file that is not located on your default device, you must specify the device name. For files on disks, you must also specify the directory where the file is cataloged.

You can use physical, logical, or generic names, described in the following sections, to refer to devices.

2.3.1 Physical Device Names

Each physical device known to the system is uniquely identified by a *physical device name*. The physical device name identifies the kind of device, for example, a storage disk or a terminal. A device name has the following format:

ddcu

The fields are as follows:

dd — Device code that represents a device type.

c — Controller designation. The controller designation, along with the unit number, identifies the location of the device within the hardware configuration of the system. Controllers are designated with alphabetic letters A through Z.

u — Unit number. The unit number, along with the controller designation, identifies the location of the device within the hardware configuration of the system. Unit numbers are decimal numbers from 0 through 65535.

The maximum length of the device name field, including the controller and the unit number, is 15 characters. When you specify a device name as part of a file specification, terminate it with a colon (:). If you do not specify a logical or physical device name, your default device name is supplied.

In addition to directory and file information, a *file specification* can include the device on which a directory and file are located. In the following example, the file STAFF_VACATIONS.TXT is located in the directory [JONES], which is located on a device with the logical name DISK2. To display the file from device DISK1, enter the following file specification:

```
$ TYPE DISK2:[JONES]STAFF_VACATIONS.TXT
```

A disk or tape must be mounted on a device in order to be recognized by the system as a *volume*. The system also recognizes *volume sets*. A volume set consists of two or more related volumes.

To access a file on a disk volume set, you have the following options:

- Specify the name of the device on which the first volume in the set is mounted. For example, if the disks DUA1 and DUA2 have been mounted as one volume set, access a file on that disk volume set by specifying DUA1 in the file specification.
- Specify the logical name assigned to the volume set when it was mounted. This is the preferred method because it allows system managers to move the volume to another device without disrupting users.

To access a file on a tape volume set, specify any device that has been allocated to that volume set. For example, if the tapes MUA1 and MUA2 have been mounted as one volume set, access a file on that tape volume set by specifying either MUA1 or MUA2 in the file specification.

2.3.2 Logical Device Names

Your system manager has probably set up logical names to represent the devices on your system. *Logical device names* can be used to equate a somewhat cryptic device name to a short, meaningful name. Use these logical names, rather than the physical device names, to refer to devices.

By using logical names, users can avoid making specific references to physical devices whose names may change. In daily system management, devices are sometimes shuffled about. You might not know when a storage disk is added to your system configuration and a frequently accessed file moved to that new disk. You continue to access the file with the same file specification because your system manager has redefined the logical name that previously pointed to one device to point to the new device.

Consequently, if your file specification contains a logical device name, you can access the file regardless of which physical device holds the disk or tape on which the file is stored. Your system manager will ensure that logical device names are always equated to the correct physical devices.

In the following example, a logical device name is used to specify the device containing the disk volume with the file STAFF_VACATIONS.TXT. Note that, like a physical device name, a logical device name must be terminated with a colon.

```
$ TYPE DISK1:[JONES]STAFF_VACATIONS.TXT
```

The VMS system also offers a special type of logical device name called a *concealed device name*. If a device has a concealed device name, the logical name (not the physical device name) will be displayed in system messages that refer to the device.

See Chapter 4 for a complete discussion of the use of logical names.

2.3.3 Generic Device Names

A *generic device name* consists of the device code and omits the specific controller or unit number. When you use a generic device name, the system locates the first available controller or device unit whose physical name satisfies the portions of the generic device name you specified.

When you use the DCL commands ALLOCATE and MOUNT, the system allows you to specify generic device names in which the controller, the unit number, or both is not specified. For example, if you enter the ALLOCATE command and specify only a device type, the ALLOCATE command locates the first available unit of that type.

For all other DCL commands, the system goes to controller A if you omit the controller designation, and to unit number 0 if you omit the unit number.

2.4 Full File Specification

As discussed in Chapter 1, a node is one of several VMS systems connected to form a computer network. If your VMS system is part of a network, the node that you access when you log in is your local node. Other nodes in the network are remote nodes. As a general user of the network, you can perform file operations on nodes other than the one at which you are logged in.

A node name can contain 1 to 6 alphanumeric characters and must contain at least one alphabetic character. A node name must always be followed by a double colon (::). You can also use a logical node name in place of the node name. For more information on logical node names, see Section 4.8.

When you add node information to the device, directory, and file information, you create a *full file specification*. A full file specification completely describes the access path the system uses to locate and identify a file. Because it describes the network node on which the file resides, a full file specification is also known as a *network file specification*.

The format for a full file specification follows:

node-name::device:[directory]filename.type;version

Assuming the file protection is set to allow remote access, the following example shows the full file specification used to display the file STAFF_VACATIONS.TXT on node HUBBUB:

```
$ TYPE HUBBUB::DISK1:[JONES]STAFF_VACATIONS.TXT
```

If you specify your local node in the file specification, DECnet–VAX logs you in over the network to perform the file operation, even though the file exists on your local node. To save time and reduce system overhead when accessing a file on your current node, omit the node name in the file specification.

The full file specification can optionally include an access control string. To indicate that you are authorized to access a file protected against network access, include an *access control string* (a 0- to 42-character string that contains a user name and password). DECnet–VAX uses this access control string to log in at the remote node. The device, directory, and file information is passed to the remote node and interpreted there.

The usual format for a full file specification that contains an access control string is as follows:

node-name"username password"::device:[directory]filename.type;version

Assume again that you are user JONES. The following example includes the access control string necessary for you to copy the file STAFF_SALARIES.TXT from your account on node HUBBUB to your default directory on another node. The asterisk at the end of the file specification is a wildcard character. Here, it instructs the system to duplicate the file name STAFF_SALARIES.TXT when that file is copied to the remote node.

```
$ COPY HUBBUB"JONES PANDEMONIUM"::DISK1:[JONES]STAFF_SALARIES.TXT *
```

If you omit the access control string, the login information sent to the remote node is determined as follows:

- If a proxy login account exists for you on the remote node, the system logs you in using that account. (A proxy login account gives access privileges on a remote node to selected users who do not have a private account on that node.)
- If no proxy login account exists, the system uses the default DECnet–VAX account for that node as specified by the local system manager.

If a file resides on a non-VMS system (that is, the file specification does not conform to VMS syntax), the name of the file as specified in this format is enclosed in a quoted string. The quotation marks prevent the local VMS system from performing syntax checking or logical name translation. In the following example, the file TEST?.DAT contains a question mark character, which is not recognized as a valid file name character in VMS:

```
$ COPY BOSTON::"TEST?.DAT" *
```

2.4.1 Using System Default Values When Specifying Files

When you enter a file specification, you can omit fields and let the system supply default values for these fields. Table 2–2 summarizes the defaults applied to each field in a file specification.

Note that the system supplies the defaults described in Table 2–2 for the first input file specification that you enter on a DCL command line.

Table 2–2: File Specification Defaults

Field	Defaults
Node	The system assumes that the default is the local system.
Device	The system uses the device (usually a disk) established at login or by the SET DEFAULT command. Devices are usually identified with logical names.
	If a physical device (ddcu) is used and a controller designation is omitted, the controller designation defaults to A. If a unit number is omitted, the unit number defaults to zero. (The ALLOCATE, MOUNT, and SHOW DEVICES commands, however, treat a device name that does not contain controller or unit numbers as a generic device name.)
Directory	The system uses the directory name established at login or by the SET DEFAULT command.
File name	No defaults are applied to the first file name in an input file specification. Most commands apply default output file names based on the file name of an input file.

Table 2-2 (Cont.): File Specification Defaults

Field	Defaults
File type	Various commands apply defaults for file types, based on the standard file type conventions summarized in Table 2-1.
File version	For input files, the system assumes the highest version number.
	For output files, if no file with the specified file name and file type exists in the current directory, the file is created with a version number of 1. However, if one or more versions do exist, the next highest version number is used.

When you enter more than one input file specification, the system applies temporary defaults for node, device, and directory names. The system uses the preceding file specification in the list that included this information. The following examples show how the system applies temporary defaults.

The following example copies the latest versions of DISK1:[JONES.TAXES.PROPERTY]DISTRICT1.DAT and DISK1:[JONES.TAXES.PROPERTY]DISTRICT2.DAT to the file AUDIT.DAT in the default directory. By default, the output (second) file specification parameter assumes the corresponding fields of the first file specification.

```
$ COPY DISK1:[JONES.TAXES.PROPERTY]DISTRICT1.DAT,DISTRICT2 AUDIT
```

When you want to specify the default file type, be sure to omit the period (which indicates a null file type).

The following example copies the files DISK1:[JONES.TAXES]BILLING.DAT and DISK1:[JONES]STAFF.DIS to DISK1:[JONES]ASSIGNMENTS.DAT. Note that the output (second) file specification parameter uses the default directory, not the directory in the first input file specification.

```
$ SET DEFAULT DISK1:[JONES]
$ COPY [.TAXES]BILLING.DAT,[]STAFF.DIS ASSIGNMENTS.DAT
```

The system applies defaults in different ways depending on the DCL command you specify. If, for example, you substitute the RENAME command for the COPY command in the previous example, you will produce one file [JONES.TAXES]ASSIGNMENTS.DAT and another [JONES]ASSIGNMENTS.DAT. See the Reference Section for more information on the defaults applied to specific DCL commands.

2.5 File Operations

File operations involve the creation, use, and deletion of files. File operations include the following:

- Displaying the contents of files
- Creating files
- Modifying files
- Copying files
- Renaming files
- Deleting files
- Printing files
- Purging files from directories
- Using wildcards

As a VMS user, you can also perform file operations over the DECnet network if you have sufficient privileges. You can display locally the contents of remote directories and files and copy files from node to node. You can print files at the remote node where they reside, copy them to a remote printing device, or copy them to the local node for printing. DCL commands permit you to access common or public directories or databases located on any node on the network. You can display their contents or print or copy the files.

See the descriptions of the DCL commands in the Reference Section for more information on specific file operations you can perform locally and over the network.

2.5.1 Using Wildcards with File Specifications

By using wildcard characters, you can apply a DCL command to multiple files rather than to one file at a time. The command applies to all files that match the portion of the file specification entered.

With many DCL commands, you can use an asterisk (*) and a percent sign (%) as a wildcard in directory names, file names, and file types. You can also use the asterisk, but not the percent sign, in version numbers.

The use of wildcard characters in DCL commands varies with the individual command. For more information on using wildcards with a particular DCL command, see the Reference Section.

2.5.1.1 The Asterisk (*) Wildcard Character

Use the asterisk wildcard character to match the following:

- An entire field, or a portion of it, in the directory, file name, and file type fields
- The entire version number field, but not a portion of it

The following example displays all versions of the file LOGIN.COM in the directory [JONES]:

```
$ TYPE [JONES]LOGIN.COM;*
```

The following example displays all versions and all file types of all files that begin with the word *STAFF* in the directory [JONES]. This would include STAFF_VACATIONS.TXT and STAFF.DIS.

```
$ TYPE [JONES]STAFF*.*;*
```

You can also use the asterisk wildcard character in a directory specification. The following example displays all versions of all files with the file type .LIS in all subdirectories one level down from [JONES]:

```
$ TYPE [JONES.*]*.LIS;*
```

You can use the asterisk in the name, type, and version fields in output file specifications. Use an asterisk in an output file specification when you want the output files to match the corresponding field in the input files.

The following example copies the latest versions of all DAT files in [JONES] to new files in [JONES] with the same name but a file type of SAV:

```
$ COPY *.DAT *.SAV
```

The following example copies the latest versions of all DAT files in [JONES] beginning with the characters 19 to new files with the same names but in the directory [SAVE]:

```
$ COPY 19*.DAT [SAVE]*.*
```

2.5.1.2 The Percent (%) Wildcard Character

The percent sign wildcard character can be used as a substitute for any single character in a file specification. You can use the percent sign in the directory, file name, and file type fields. You cannot, however, use the percent sign in the version number field.

The following example displays the latest versions of all DAT files whose names begin with DISTRICT:

```
$ TYPE [JONES.TAXES.PROPERTY]DISTRICT%.DAT
```

This display would include the files DISTRICT1.DAT, DISTRICT2.DAT, and DISTRICT3.DAT. The file DISTRICT4_5.DAT would not be displayed because it has more than one character after DISTRICT, nor would the file DISTRICT.DAT be displayed. The percent sign replaces one character position in a field, but there must be a character to replace.

2.5.2 Displaying the Contents of Files

You can display the contents of files on your terminal screen by using the TYPE command or by invoking an interactive text editor with the /READ_ONLY qualifier. The following example displays the file STAFF_VACATIONS.TXT:

```
$ TYPE STAFF_VACATIONS.TXT
```

The following example displays the file COMPANY_HOLIDAYS.TXT, which is located on remote node CHAOS:

```
$ TYPE CHAOS::DISK2:[PUBLIC]COMPANY_HOLIDAYS.TXT
```

If more than one file is listed in the TYPE command, the files are displayed in the order specified; if wildcard characters are used, the files are displayed in alphabetical order.

To stop the scrolling of the text on the screen temporarily, press the HOLD SCREEN key (F1 on VT200- and VT300-series terminals); to resume scrolling, press the HOLD SCREEN key again. To stop the display and return to DCL command level, press CTRL/Y or CTRL/O.

If you specify the /PAGE qualifier to the TYPE command, you can view one screen at a time. The system prompts you to press RETURN when you want to see the next screen.

By invoking an interactive text editor (for example, EVE or EDT) with the /READ_ONLY qualifier, you can use interactive editing commands to move around in a file and search for specific sequences of characters. The /READ_ONLY qualifier prevents you from modifying the file as you display it. Control characters are displayed rather than being interpreted when you use /READ_ONLY, however. For example, the form-feed character appears as <FF> rather than producing a form feed.

2.5.3 Creating and Modifying Files

The most versatile interactive tool for creating and modifying files is the interactive editor. EVE and EDT are two such editors; VMS supports several others. See Chapter 8 for a description of the EVE and EDT editors.

You can also create and modify files by using the DCL commands CREATE, COPY, and RENAME. The CREATE command creates a text file. You enter the CREATE command and then type lines of text, as shown in the following example:

```
$ CREATE POUND.LIS
Tag #23, Elmer Doolittle, notified
Tag #37, James Watson, notified
No tag, light brown, 30 lbs., looks part beagle
CTRL/Z
```

Pressing CTRL/Z signals the end of the file and returns you to DCL command level. You cannot modify a file with the CREATE command. Once you have pressed RETURN, you cannot return to a previous line to modify a word.

The COPY command duplicates the contents of the old file in a new file. The following example copies FEES.DAT to RECORDS.DAT in the default directory:

```
$ COPY FEES.DAT RECORDS
```

The COPY command can duplicate many files at a time. The following example copies all TXT files in the default directory to another directory:

```
$ COPY *.TXT;* [SAVETEXT]*.*;*
```

The COPY command can concatenate files. The following example appends FEES1.DAT to FEES.DAT (forming a new version of FEES.DAT) in your default directory:

```
$ COPY FEES.DAT,FEES1.DAT FEES.DAT
```

Use the COPY command to copy files from another node to your node. The following example copies the latest version of all files in DISK2:[PUBLIC] on node CHAOS to files with the same names in your default directory:

```
$ COPY CHAOS::DISK2:[PUBLIC]*.*  *
```

Use the COPY command to copy files from your node to another node. The following example copies the latest version of all files in your default directory to files with the same names in the directory DISK2:[STAFF_BACKUP] on node CHAOS:

```
$ COPY *.* CHAOS::DISK2:[STAFF_BACKUP]
```

If you receive a protection violation or DECnet–VAX error message when you attempt to copy a file across systems, you have two recourses:

- If you own the file, you can send it to a user account on the other node with the Mail Utility.
- You can follow the node name in the file specification with an access control string.

Use the /SINCE qualifier with the COPY command to select only those files that meet the specified criterion. The following example copies to the default directory only those files in the directory [JONES.LICENSES.DOG] that have been modified since December 31, 1988:

```
$ COPY/SINCE=31-DEC-1988/MODIFIED [JONES.LICENSES.DOG]*.* *
```

Use the RENAME command to give the file a new name and optionally locate it in a different directory. The following example gives the file FEES.DAT the new name RECORDS.DAT and moves it from the default directory to another directory:

```
$ RENAME FEES.DAT;4 [SAVETEXT]RECORDS.DAT
```

Note that after being renamed, the file FEES.DAT;4 no longer exists in the default directory. When you use the RENAME command, the input and output locations must be on the same device.

2.5.4 Deleting Files

The DELETE command removes files from directories and releases the disk space they occupy for use by other files. The DELETE command requires you to specify a version number or the asterisk wildcard character in each file specification. The following example deletes version 17 of POUND.LIS:

```
$ DELETE POUND.LIS;17
```

The following example deletes versions 16 and 17 of POUND.LIS:

```
$ DELETE POUND.LIS;16,;17
```

The following example deletes all versions of POUND.LIS:

```
$ DELETE POUND.LIS;*
```

When you delete many files with wildcard characters, you should confirm each deletion by specifying the /CONFIRM qualifier, as shown in the following example:

```
$ DELETE/CONFIRM *.*;*
DISK1:[JONES.LICENSES.DOG]FEES.DAT;4, delete? [N]:
DISK1:[JONES.LICENSES.DOG]FEMALE.LIS;6, delete? [N]:
DISK1:[JONES.LICENSES.DOG]MALE.LIS;3, delete? [N]:
DISK1:[JONES.LICENSES.DOG]POUND.LIS;17, delete? [N]:
```

Similarly, you may want to display the names of files as they are deleted. You can do this by specifying the /LOG qualifier with the DELETE command, as shown in the following example:

```
$ DELETE/LOG *.LIS;*
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]FEMALE.LIS;6 deleted (35 blocks)
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]MALE.LIS;3 deleted (5 blocks)
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]POUND.LIS;17 deleted (9 blocks)
```

The PURGE command deletes all but the latest version of the specified file (or all files) in the default directory or any other specified directory. Purging sequential files after updating them enables you to retain more free space on your disk volumes.

The following example deletes all but the latest two versions of each file in your default directory:

```
$ PURGE/KEEP=2
```

2.5.5 Printing Files

The PRINT command places your print job (all the files to be printed) in a list of jobs to be printed called a *print queue*. Print queues can be one of the following types of queues:

- Print queue—A queue assigned to a specific print device.
- Terminal queue—A print queue assigned to a hardcopy terminal that is being used solely as a printer (not interactively).
- Generic queue—A queue that distributes the processing of jobs to printers with similar characteristics. Jobs submitted to a generic queue are held in that queue until one of the assigned printer queues becomes available.

To print a file or files, use the PRINT command. The following example places a print job containing three files in the default print queue, SYS$PRINT.

```
$ PRINT POUND,MALE,FEES.DAT
Job POUND (queue SYS$PRINT, entry 202) started on SYS$PRINT
```

The file types of the files named in the PRINT command default to LIS or the last explicitly named file type; thus, the preceding example queues POUND.LIS, MALE.LIS, and FEES.DAT to SYS$PRINT. The system displays the job name (POUND), the queue name (SYS$PRINT), the job number (202), and indicates whether the job has started or is pending. By default, the job name is the name of the first (or only) file specification in the PRINT command. Once a job is submitted to a queue, you reference it using the job number. Once the job is queued, it will be printed when no other jobs precede it in the queue and when the printer is physically ready to print.

A print queue can execute only one job at a time. Print jobs are scheduled for printing according to their priority, and the job with the highest priority is printed first. If more than one job exists with the same priority, the smallest job is usually printed first. Jobs of equal size having the same priority are selected for printing according to their submission time.

The default print queue, SYS$PRINT, is usually initialized and started as part of the site-specific system startup procedures. The SHOW QUEUE command displays the queues that are initialized at your site. The SHOW ENTRY command displays the status of your print jobs, as shown in the following example:

```
$ SHOW ENTRY

   Jobname    Username   Entry   Blocks   Status
   -------    --------   -----   ------   ------
   POUND      JONES        202       38   Printing
On printer queue SYS$PRINT
```

Specify the USERNAME parameter to the SHOW ENTRY command to see jobs queued by other users. Use the ENTRY-NUMBER parameter to the DELETE/ENTRY command to delete your job from the queue, as shown in the following example:

```
$ DELETE/ENTRY=202
```

You can print a file on another system by copying that file to the remote node and specifying the /REMOTE qualifier to the PRINT command. The following example copies the file COMPANY_HOLIDAYS.TXT from your local node to the remote node CHAOS and queues the file for printing to the default system print queue (SYS$PRINT) on node CHAOS. The asterisk at the end of the file specification is a wildcard character. Here, it instructs the system to duplicate the file name COMPANY_HOLIDAYS.TXT when that file is copied to the remote node.

```
$ COPY COMPANY_HOLIDAYS.TXT CHAOS"JONES PANDEMONIUM"::DISK2:[JONES]*
$ PRINT/REMOTE CHAOS::DISK2:[JONES]COMPANY_HOLIDAYS.TXT
```

In the previous example, an access control string was specified to indicate that you are authorized to copy files to the directory [JONES] on node CHAOS. However, if you have a proxy account on that remote node, the access control string is unnecessary. (See Section 2.4 for more information about proxy accounts.)

Note that not all qualifiers to the PRINT command are compatible with the /REMOTE qualifier. For example, you cannot queue a job to a specific print queue; all jobs are queued to the default system print queue (SYS$PRINT). See the description of the /REMOTE qualifier to the DCL command PRINT in the Reference Section for a list of PRINT qualifiers compatible with /REMOTE.

DCL Commands That Control Print Jobs

The DCL commands listed in the following table allow you to control print jobs in various ways. For example, you can specify the number of copies printed or you can request that the system notify you when your print job is complete. For more information on any of these commands, see the descriptions of the DCL commands in the Reference Section.

Print Operations	Print Job Commands and Qualifiers
Number of copies	
By job	PRINT/JOB_COUNT=n[1]
By file	PRINT/COPIES=n[1]
Specified file only	file-spec/COPIES=n[1]
Number of pages	PRINT/PAGES=[1]
Print features	
Flag pages	PRINT/FLAG=[1]
Type of forms (paper)	PRINT/FORM=[1]
Special features	PRINT/CHARACTERISTICS=[1]
Double-spacing	PRINT/SPACE[1]
Page heading	PRINT/HEADER[1]
Notification of job execution	PRINT/NOTIFY
Delay execution of a job	
For a specified time	PRINT/AFTER
Indefinitely	PRINT/HOLD
Release a delayed job	SET QUEUE/ENTRY/RELEASE
Display your print jobs	SHOW ENTRY
Stop a print job	
Delete job	DELETE/ENTRY=job-number
Stop currently printing job and begin printing the next job in the queue	STOP/ABORT
Stop currently printing job and requeue it for printing	STOP/REQUEUE

[1]Parallel qualifiers for the SET QUEUE/ENTRY command allow you to specify these operations for print jobs that are already queued but not yet printing.

2.6 Device and Directory Operations

To access files on your system, you need to know how to navigate through many directory structures. Because directories reside on devices, you also need to know how to work with devices other than your default device.

Device and directory operations include the following:

- Displaying directories
- Creating directories
- Deleting directories
- Setting a default device
- Setting a default directory

- Searching the directory structure with wildcards

2.6.1 Displaying Directories

The DCL command DIRECTORY displays the names of the files in a directory. The following example lists the files in [JONES]:

```
$ DIRECTORY/COLUMNS=1

Directory DISK1:[JONES]

LICENSES.DIR;1
LOGIN.COM;3
LOGIN.COM;4
STAFF.DIS;3
STAFF_VACATIONS.TXT;2
TAXES.DIR;1

Total of 5 files.
```

The display shows us that [JONES] contains two subdirectories—[JONES.LICENSES] and [JONES.TAXES]—and four nondirectory files—STAFF.DIS, STAFF_VACATIONS.TXT, and two versions of LOGIN.COM. The following example (assuming that the default directory remains [JONES]) lists the contents of the subdirectory [JONES.LICENSES]. Note that if you want to move one level down the directory structure, you need specify only the subdirectory name, preceded by a period, to which you want to move.

```
$ DIRECTORY/COLUMNS=1 [.LICENSES]

Directory DISK1:[JONES.LICENSES]

MAILING.LIS;6
TOTAL.DAT;2
DEPT.DAT;3
DOG.DIR;1
MARRIAGE.DIR;1

Total of 6 files.
```

If you have sufficient privileges, you can display the contents of the master file directory. To do so, specify [000000] as the *file-spec* parameter to the DIRECTORY command or search up one level from a top level directory using the [-] wildcard (described in Section 2.6.6.2). Note that nine of the files contained in [000000] are structure files, which are special files created and reserved by the system that must not be deleted. Note also that your system disk contains several directories with files that provide data required to run VMS and allow you to run command images and execute command procedures.

See the *VMS System Manager's Manual* for more information about user privileges and system directories and files.

2.6.2 Creating Directories

The CREATE/DIRECTORY command creates a subdirectory, as shown in the following example:

```
$ CREATE/DIRECTORY [JONES.LICENSES]
```

If your current default directory is [JONES], specify the following:

```
$ CREATE/DIRECTORY [.LICENSES]
```

Note that you must have SYSPRV privilege to create a top level directory. See the *VMS System Manager's Manual* for a discussion of user privileges.

2.6.3 Deleting Directories

You cannot delete a directory that contains files. Before deleting a directory or subdirectory, make sure it is empty by entering the DIRECTORY command, as shown in the following example:

```
$ SET DEFAULT [JONES.LICENSES]
$ DIRECTORY
  No files found.
```

If the directory contains any files, copy or rename them to another directory (if you want to save them) and delete them from the directory of interest. If the directory contains subdirectories, examine those subdirectories (copying and deleting their files) and delete the subdirectories.

To delete a directory, move to the directory one level above the directory you want to delete. This means that if you want to delete [JONES.LICENSES], you should set default to [JONES]. Remember that the subdirectory [JONES.LICENSES] exists as a file named LICENSES.DIR;1 in the directory [JONES]. You delete a directory by deleting the file that points to that directory.

Because a directory file is created without delete access to prevent accidental deletion of the directory, you must change the file protection to allow delete access before you can delete that directory file. (See Section 7.2 for more information about file protection.) The following example shows how to delete the subdirectory [JONES.LICENSES]:

```
$ SET DEFAULT [JONES]
$ SET PROTECTION=OWNER:D LICENSES.DIR
$ DELETE LICENSES.DIR;1
```

The directory files (for example, JONES.DIR;1) in the master file directory require SYSPRV privilege to delete. See the *VMS System Manager's Manual* for a discussion of user privileges.

2.6.4 Setting a Default Directory

Ascend and descend the directory hierarchy by setting default to a different directory with the DCL command SET DEFAULT. The default remains in effect until you enter another SET DEFAULT command.

The following example sets default to the directory [JONES] and displays the file STAFF_VACATIONS.TXT:

```
$ SET DEFAULT [JONES]
$ TYPE STAFF_VACATIONS.TXT
```

Subdirectories are specified by concatenating the subdirectory name to the name of the directory one level above it. The following example displays the file BILLING.DAT located in the subdirectory [JONES.TAXES]:

```
$ SET DEFAULT [JONES.TAXES]
$ TYPE BILLING.DAT
```

When you move from your current default directory to a subdirectory one level below, you can omit the current directory's name in the file specification. By default, the system assumes the current directory. In the following example, the current default directory is [JONES]:

```
$ SET DEFAULT [.TAXES]
$ TYPE BILLING.DAT
```

You can display the current default directory by entering the command SHOW DEFAULT, as shown in the following example:

```
$ SHOW DEFAULT
  DISK1:[JONES.TAXES]
$ SET DEFAULT [PUBLIC]
$ SHOW DEFAULT
  DISK1:[PUBLIC]
```

2.6.5 Setting a Default Device

Section 2.6.4 describes how to set a default directory with the DCL command SET DEFAULT. You can also use the SET DEFAULT command to change the default device. The default remains in effect until you enter another SET DEFAULT command or log out.

The following example shows how to change the default device:

```
$ SHOW DEFAULT
  DISK1:[JONES]
$ SET DEFAULT DISK2:[GROUP]
$ SHOW DEFAULT
  DISK2:[GROUP]
```

You can specify the device to which you want to set default without including the directory in the command. In the following example, the directory [JONES] is assumed and exists on DISK1 and DISK2:

```
$ SHOW DEFAULT
  DISK1:[JONES]
$ SET DEFAULT DISK2:
$ SHOW DEFAULT
  DISK2:[JONES]
```

Note that VMS allows you to set default to a nonexistent disk or directory. If you find yourself in a nonexistent disk or directory and cannot carry out a desired operation, simply set default to an existing disk or directory and continue your task.

2.6.6 Searching the Directory Structure with Search Wildcards

From any point in a directory structure, you can refer to another directory or subdirectory in the structure. Do this by specifically naming the directory or subdirectory you want or by using the ellipsis (. . .) and hyphen (-) wildcard characters.

2.6.6.1 The Ellipsis (. . .) Wildcard Character

Use the ellipsis (. . .) to search down into the directory hierarchy. To search the current directory and all the subdirectories below it, use the ellipsis by itself. The following command searches the current default directory and all subdirectories below it:

```
$ DIRECTORY [...]
```

Assuming the current directory is [JONES], the following command displays the latest versions of all files named FEES.DAT in [JONES] and all subdirectories under [JONES]:

```
$ TYPE [JONES...]FEES.DAT
```

If you begin the directory specification with an ellipsis, the search begins from your current directory. Assuming the current default directory is [JONES], the following command searches all subdirectories that end in .SALES and displays the latest versions of the file FEDERAL.LIS:

```
$ TYPE [...SALES]FEDERAL.LIS
```

Assuming the current directory is [JONES], the following command displays the latest versions of all files named DEPT.DAT in [JONES] and all subdirectories under [JONES]:

```
$ TYPE [...]DEPT.DAT
```

However, if you begin the directory specification with a period, only the subdirectory that is one level lower than the current directory is searched. Assuming the current directory is [JONES], the following command searches only the [.LETTERS] subdirectory that is one level lower than [JONES] for the file INVITATION.TXT. The subdirectory [JONES.LETTERS] is searched, but [JONES.WORK.LETTERS] is not:

```
$ TYPE [.LETTERS]INVITATION.TXT
```

Assuming the current directory is [JONES], the following command displays the latest versions of all files named DEPT.DAT in the [.LICENSES] subdirectory under [JONES] and all subdirectories under the [.LICENSES] subdirectory:

```
$ TYPE [...LICENSES...]DEPT.DAT
```

To search all top level directories and their subdirectories from wherever you are in the directory structure, use an asterisk (*) followed by an ellipsis (. . .). The following command (which requires READALL privilege) searches as many as eight levels of directory names (the top level directory and seven subdirectories), if they exist. It does not search the MFD.

```
$ DIRECTORY [*...]
```

2.6.6.2 The Hyphen (-) Wildcard Character

The hyphen (-) wildcard character permits you to move up through the directory structure. Each hyphen refers to the directory one level up from the current one. You can follow the hyphens with directory and subdirectory names to move down the directory structure on another path.

If the current directory is [JONES.LICENSES], the following command displays the latest version of STAFF.DIS in [JONES]:

```
$ TYPE [-]STAFF.DIS
```

If your current directory is [JONES.LICENSES], the following command displays the latest version of BILLING.DAT in [JONES.TAXES]:

```
TYPE [-.TAXES]BILLING.DAT
```

You can specify more than one hyphen. The following command moves you up two levels in the directory hierarchy. From there, you are placed in the top level directory [SMITH].

```
$ SET DEFAULT [--SMITH]
```

If you enter so many hyphens that you point above the master file directory (MFD), the system displays an error message.

The following example uses the BACKUP command to copy all files in [JONES.TAXES] to a directory named [AUDIT], and all the files in any subdirectories under [JONES.TAXES] to corresponding subdirectories under [AUDIT]:

```
$ BACKUP [JONES.TAXES...]*.*.* [AUDIT...]*.*.*
```

The trailing ellipsis in the output specification lets you move the entire third level directory structure from the input directory to the second level of the output directory. Unlike the COPY command, the BACKUP command preserves the input directory structure in the output it creates.

2.6.6.3 Using Wildcards to Copy a Directory Structure

By including asterisk (*) and ellipsis (. . .) wildcards in output directory specifications, you can do the following:

- Duplicate an entire input directory structure
- Move files from one directory structure into another directory structure at the same or at a different level

Each wildcard character in an output directory specification refers to a corresponding directory level in the input specification. An output specification may contain only wildcards, or it may contain a combination of wildcards and directory names. If directory names are used, they must always precede any wildcards that are included.

Use the asterisk when you want a particular level in the output directory specification to match a level indicated by a wildcard in the input specification. For example:

```
$ BACKUP [JONES.*]*.*;* [SCHULTZ.SAVE.*]*.*;*
```

In the previous example, the BACKUP command copies all files from any subdirectories under [JONES] to any corresponding subdirectories under [SCHULTZ.SAVE]. For example, all files in the subdirectory [JONES.TAXES] are copied to the subdirectory [SCHULTZ.SAVE.TAXES]. Notice that the single asterisk in the output directory specification refers to the first subdirectory level in the input directory that contained a wildcard.

Use the ellipsis when you want the output directory specification to follow the same structure downwards as the input directory from the first level that contained a wildcard. For example:

```
$ BACKUP [JONES.TAXES...]*.*;* [AUDIT...]*.*;*
```

In the previous example, the BACKUP command copies all files in [JONES.TAXES] to a directory named [AUDIT], and all the files in all subdirectories under [JONES.TAXES] to corresponding subdirectories under [AUDIT]. The trailing ellipsis in the output specification lets you move the entire third level directory structure from the input directory to the second level of the output directory.

For output directory specifications, a trailing asterisk and ellipsis are mutually exclusive when they follow a specific directory name. Therefore, output directory specifications such as [USER.* . . .] and [USER . . . *] are invalid. However, [* . . .] is valid, because the asterisk wildcard is used in place of a directory name.

You can move an entire input directory structure to an output directory structure. The two ways to do this are as follows:

```
$ BACKUP DISK1:[JONES...]*.*;* DISK2[*]*.*;*
```

or

```
$ BACKUP DISK1:[JONES...]*.*;* DISK2[*...]*.*;*
```

These commands let you move all the files in the [JONES] directory structure on the disk DISK1 to the [JONES] directory structure on disk DISK2, from the top level directory down through the entire structure.

Chapter 3

Working with Processes

The environment in which you interact with the system is called a *process*. A process contains identification and status information that the system needs to execute programs for you. Within a process, programs execute one at a time in the order in which they are invoked.

You can place your process into hibernation and create a second process called a *subprocess* under your user name. You can interact with the system and log out of that subprocess to return to the original process.

A program executes within the context of the process that invokes it. Some programs are system programs that control the flow of events within the process. For example, when you log in, your process is under the control of the system program SYS$SYSTEM:LOGINOUT.EXE. When you work at DCL level, your process is under the control of SYS$SYSTEM:DCL.EXE.

A command procedure is a file that contains a list of DCL commands. Complex command procedures resemble programs written in high-level programming languages. In this sense, command procedures provide a way to write programs in DCL.

You can submit programs and command procedures for execution as batch jobs, which you submit to the system as separate processes. Batch jobs allow you to continue to work interactively with the system while the program or procedure executes as another process under your user name.

3.1 Processes and the User Environment

Each user on the system is associated with a process, which is a special environment created by the system that makes interaction with the system possible. A process has a beginning and the end; for example, the system creates a process for you when you log in and deletes that process when you log out. A process contains all the information that the system needs to execute programs. It is within your process that the system executes your programs (also called images or executable images) one at a time.

A process can be a *detached process* (a process that is independent of other processes) or a *subprocess* (a process that is dependent on another process for its existence and resources). Your main process, also called your parent process, is a detached process.

The system creates a process for you when you do one of the following:

- Log in—The system creates a process for each interactive user.
- Submit a batch job—The system creates a process for each batch job. When the batch job is completed, the system deletes the process. Section 3.1.4 discusses batch jobs.
- Spawn a subprocess—The system creates a process when you use the SPAWN command. Section 3.1.3 describes subprocesses.
- Run a program using either the /DETACHED or the /UIC=uic qualifier. Section 3.1.1 describes programs.

The system also creates special system processes to perform various functions. The DCL command SHOW SYSTEM displays both user and system processes.

The following list summarizes the *process context*. Certain characteristics, such as the privileges, symbols, and logical names enabled in your process, collectively create the process context. Use the DCL command SHOW PROCESS/ALL to examine your process context.

```
31-DEC-1988 13:30:37.12 ❶                    User: CLEAVER ❷
Pid: 24E003DC ❸     Proc. name: CLEAVER_1 ❹     UIC: [DOC,CLEAVER] ❺
Priority:   4 ❻     Default file spec: DISK1:[CLEAVER] ❼

Process Quotas: ❽
 Account name: DOC
 CPU limit:                        Infinite  Direct I/O limit:          18
 Buffered I/O byte count quota:       31808  Buffered I/O limit:        25
 Timer queue entry quota:                10  Open file quota:           57
 Paging file quota:                   22276  Subprocess quota:           4
 Default page fault cluster:             64  AST quota:                 38
 Enqueue quota:                         600  Shared file limit:          0
 Max detached processes:                  0  Max active jobs:            0
```

```
Accounting information: ❾
 Buffered I/O count:          140  Peak working set size:        383
 Direct I/O count:              7  Peak virtual size:           2336
 Page faults:                 304  Mounted volumes:                0
 Images activated:              1
 Elapsed CPU time:        0 00:00:00.55
 Connect time:            0 00:00:22.76

Process privileges: ❿
 GROUP                 may affect other processes in same group
 TMPMBX                may create temporary mailbox
 OPER                  operator privilege
 NETMBX                may create network device

Process rights identifiers: ⓫
 INTERACTIVE
 LOCAL
 SYS$NODE_AJAX

Process Dynamic Memory Area ⓬
    Current Size (bytes)          25600   Current Total Size (pages)      50
    Free Space (bytes)            19592   Space in Use (bytes)          6008
    Size of Largest Block         19520   Size of Smallest Block          24
    Number of Free Blocks             3   Free Blocks LEQU 32 Bytes        1

Processes in this tree: ⓭
CLEAVER
  CLEAVER_1 (*)
```

❶ Current date and time—The date and time when the SHOW PROCESS/ALL command is executed.

❷ User name—The user name assigned to the account that is associated with the process.

❸ Process identification number (PID)—A unique number assigned to the process by the system. The SHOW PROCESS command displays the PID as a hexadecimal number.

❹ Process name—The name assigned to the process. Since process names are unique, the first process logged in under an account is assigned the user name, and subsequent processes logged in under the same account are assigned the terminal name. You can change your process name with the DCL command SET PROCESS/NAME.

❺ User identification code (UIC)—The group and member numbers (or letters) assigned to the account that is associated with the process (for example, [PERSONNEL,RODGERS]). Part of your UIC identifies the group to which you belong. Within a group, users are allowed to share files or system resources more freely than between groups.

❻ Priority—The current priority of the process.

❼ Default file specification—The current device and directory. Change your current defaults with the DCL command SET DEFAULT.

❽ Process quotas—The quotas (limits) associated with the process. Examine these quotas with the /QUOTAS or /ALL qualifiers of the SHOW PROCESS command.

❾ Accounting information—The continuously updated account of the process's use of memory and CPU time. Examine this information with the /ACCOUNTING or /ALL qualifiers of the SHOW PROCESS command.

❿ Process privileges—The privileges granted to your processes. Privileges restrict the performance of certain system activities to certain users. Examine your privileges with the /PRIVILEGES or /ALL qualifiers of the SHOW PROCESS command.

⓫ Process rights identifiers—System-defined identifiers that are used in conjunction with access control list protection. Identifiers provide the means of specifying the users in an access control list. An access control list is a security tool that defines the kinds of access to be granted or denied to users of an object, such as a file, device, or mailbox. (See Chapter 7 for more information about identifiers and access control lists.)

⓬ Process dynamic memory area—The process's current use of dynamic memory. Dynamic memory is allocated by the system to an image when that image is executing. When that memory is no longer needed by one process, the system allocates it to another process. Examine this information with the /MEMORY or /ALL qualifiers of the SHOW PROCESS command.

⓭ Processes in this tree—A list of subprocesses belonging to the parent process. An asterisk appears after the current process. Examine this with the DCL SHOW PROCESS/SUBPROCESSES or /ALL command.

3.1.1 Programs

A program, also called an *image* or *executable image*, is a file that contains instructions and data in machine-readable format. Image files can be VMS- or user-supplied and usually have a file type of EXE. You cannot examine an image file with the DCL commands TYPE, PRINT, or EDIT because image files do not consist of ASCII characters. (Text files contain ASCII characters, which are a standard method of representing the alphabet, punctuation marks, numerals, and other special symbols.)

A program can be either a command image or a noncommand image as follows:

- Command image—A command image is a program associated with and invoked by a DCL command. For example, when you type the DCL command COPY, the system executes the program SYS$SYSTEM:COPY.EXE. COPY.EXE is a command image. A system directory named SYS$SYSTEM contains a number of command image files, most of which are VMS-supplied. Use the DCL command DIRECTORY SYS$SYSTEM to examine this system directory.

- Noncommand image—A noncommand image is a program not associated with a DCL command. To invoke a noncommand image, name the file containing the program as the parameter to the RUN command.

Executing Programs Across the Network

Because of support provided by DECnet–VAX, programs can execute across the network as if they were executing locally. Because DECnet–VAX is integrated within the VMS operating system, it is easy to write programs that access remote files. To access a remote file in an application program, you need only include in your file specification the name of the remote node and any required access control information.

Task-to-task communications, a feature common to all DECnet implementations, allows two application programs running on the same or different operating systems to communicate with each other regardless of the programming languages used. Examples of network applications are distributed processing applications, transaction processing applications, and applications providing connection to servers.

3.1.2 Command Procedures

A command procedure is a file that contains a list of DCL commands. When you execute a command procedure, DCL reads the command file and executes the commands it contains. Command procedures can be executed as interactive or batch processes. If you use command procedures that require lengthy processing time (for example, the compilation or assembly of large programs), submit these procedures as batch jobs so you can continue to use your terminal interactively.

When you submit a command procedure for batch execution, the system creates a detached process using your account and process characteristics. The system runs the job from that process and deletes the process when the job is completed.

You can use command procedures to automate sequences of commands that you enter frequently. For example, if you always examine the contents of a directory immediately after setting default to it, you can design a command procedure that issues the appropriate commands to display the directory's contents. A command procedure might contain the following commands to set default to the ACCOUNT subdirectory and display the subdirectory's contents. (Exclamation points delimit comments in command procedures; DCL ignores everything to the right of the exclamation point when processing the line.)

```
$ ! DISK1:[ADAMS]ACCOUNTD.COM
$ !
$ SET DEFAULT DISK1:[ADAMS.ACCOUNT]
$ DIRECTORY
```

To execute a command procedure interactively, type the @ command followed by the procedure's file specification. To execute the command procedure in the previous example, enter @DISK1:[ADAMS]ACCOUNTD (or @ACCOUNTD if your current disk and directory are DISK1:[ADAMS]).

Chapter 6 discusses command procedures in greater detail.

3.1.3 Subprocesses

The SPAWN command enables you to create a subprocess of your current process. Within this subprocess, you can interact with the system and log out of the subprocess to return to your parent process, or switch between your parent process and subprocesses. Only one of your processes is executing at any time.

Each user on the system is represented by a *job tree*. A job tree is a hierarchy of all your processes and subprocesses, with your main process at the top. A subprocess is dependent on the parent process and is deleted when the parent process exits. By default, the subprocess assumes the name of the parent process followed by an underscore and a unique number. For example, if the parent process name is DOUGLASS, the subprocesses are named DOUGLASS_1, DOUGLASS_2, and so on, forming a tree of subprocesses.

Typically, you use a subprocess in one of the following two ways:

- To interrupt a task, perform a second task, then return to the original task—Because SPAWN is a built-in command (listed in Chapter 1), you can use CTRL/Y to interrupt one task, spawn a subprocess to perform a second task, exit from the subprocess, and then enter the CONTINUE command to return to the original task. By default, when you create a subprocess, the parent process hibernates, and you are given control at DCL level within the subprocess. Your default directory is the current directory of the parent process. (If you interrupt the EDT editor, enter the CONTINUE command and press CTRL/W to refresh the screen.)

- To perform a second task while continuing to work on your original task—You can do so by creating the subprocess with the SPAWN/NOWAIT command. Use the SPAWN/NOWAIT command only to execute commands that do not require input; SPAWN/NOWAIT generates a noninteractive, batch-like subprocess.

  Because both the parent and the subprocess are executing concurrently, both attempt to control the terminal. To prevent conflicts, also specify the following:

  - /OUTPUT qualifier—Indicates that the subprocess should write output to a specified file rather than to the terminal.

  - SPAWN command parameter or /INPUT qualifier—Indicates that the subprocess should execute the specified commands rather than reading input from the terminal.

  When you specify the /INPUT qualifier of the SPAWN command, the subprocess is created as a noninteractive process that exits upon encountering a severe error or an end-of-file indicator. At DCL level, CTRL/Z is treated as an end-of-file indicator.

In the following example, a command image (the TYPE command) is interrupted with CTRL/Y and a subprocess is spawned:

```
$ TYPE MICE.TXT
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
    .
    .
    .
CTRL/Y
$ SPAWN
%DCL-S-SPAWNED, process DOUGLASS_1 spawned
%DCL-S-ATTACHED, terminal now attached to process DOUGLASS_1
$ MAIL
MAIL>
    .
    .
    .
MAIL> EXIT
$ LOGOUT
  Process DOUGLASS_1 logged out at 31-DEC-1988 12:42:12.46
%DCL-S-RETURNED, control returned to process DOUGLASS
$ CONTINUE
Once inside, they may gnaw through electrical wires and raid
your food. Because mice reproduce so quickly, what started
as one or two mice can quickly become an invasion.  If you seal
the cracks and holes on the exterior of your foundation, you can
prevent these rodents from ever getting in.
```

Because each process you create is unique, commands executed in one process do not usually affect any other process. However, because control of the terminal passes between processes, commands that affect the terminal characteristics (for example, SET TERMINAL) affect any process controlling that terminal. For example, if one process inhibits echoing and exits without restoring it, echoing remains inhibited for the next process that gains control of the terminal. Reset any altered terminal characteristics with the SET TERMINAL command.

3.1.3.1 Exiting from a Subprocess

To exit from a subprocess created by SPAWN, use one of the following commands:

- LOGOUT—When you exit from a subprocess with the LOGOUT command, the subprocess is deleted (along with any subprocesses that it created), and you are returned to the parent process.
- ATTACH—When you exit from a subprocess with the ATTACH command, the subprocess hibernates, and control of your terminal is transferred to the specified process. (You must specify either a process name as a parameter to the ATTACH command or a process identification number (PID) as a value of the /IDENTIFIER qualifier of the ATTACH command.) The following example shows how to exit from the subprocess DOUGLASS_1 and attach to the process DOUGLASS:

```
$ ATTACH DOUGLASS

%DCL-S-RETURNED, control returned to process DOUGLASS

$ SHOW PROCESS

26-APR-1988 10:34:58.50   VTA303              User: DOUGLASS
Pid: 25C002B4   Proc. name: DOUGLASS          UIC: [200,200]
Priority:   4   Default file spec: SYS$SYSDEVICE:[DOUGLASS]

Devices allocated: $11$VTA303:
```

3.1.3.2 Subprocess Context

By default, a subprocess inherits the following items from the parent process: defaults, privileges, symbols, logical names, control characters, message format, verification state, and key definitions. The environment that these items collectively create is called the *process context*. The following items, however, are not inherited from the parent process:

- Process identification number (PID)—The system assigns each created subprocess a unique process identification number.
- Process name—By default, the subprocess name consists of the name of the parent process followed by an underscore and an integer. Use the /PROCESS qualifier of the SPAWN command to specify a process name other than the default. A process name must be unique.
- Created commands—Commands that are defined by a parent process using the SET COMMAND command are not copied to a subprocess. To use a created command in a subprocess, you must use SET COMMAND to create that command for the subprocess.
- Authorize privileges—When you spawn to a subprocess, the process context contains the privileges currently enabled, not the privileges that you may be authorized to enable. For example, if you spawn to a subprocess while in MAIL and want to perform a privileged operation, you need to have already set the proper privilege in the parent process.

You can use the following SPAWN qualifiers to prevent the subprocess from inheriting a number of these items:

Qualifier	Items Inhibited or Changed
/CARRIAGE_CONTROL, /PROMPT	DCL prompt
/NOCLI	CLI (command language interpreter; DCL by default)
/NOKEYPAD	Keypad definitions
/NOLOGICAL_NAMES	Logical names
/NOSYMBOL	Symbols

The /SYMBOL and /LOGICAL_NAMES qualifiers do not affect system-defined symbols (such as $SEVERITY and $STATUS) or system-defined logical names (such

as SYS$COMMAND and SYS$OUTPUT). Symbols are described in Chapter 5. See Chapter 4 for more information about logical names.

Since copying logical names and symbols to a subprocess can be time-consuming (a few seconds), you may want to use the /NOLOGICAL_NAMES and /NOSYMBOL qualifiers to the SPAWN command unless you plan to use the logical names or symbols in the subprocess. If you use subprocesses frequently, the ATTACH command provides the most efficient way to enter and exit a subprocess. This method allows you to transfer control quickly between the parent process and subprocess rather than repeatedly waiting for the system to create a new subprocess for you.

3.1.4 Batch Jobs

Usually you use VMS in interactive mode. When you work interactively with VMS, the system must expend resources waiting for your input. If the system is heavily loaded with other jobs, your interaction with VMS is slowed. Also, when you run a long job interactively, your terminal is unavailable to you until the job is completed.

A batch job consists of one or more programs or command procedures that execute as a detached process without user interaction. A batch job allows you to use your terminal for other work while the system executes your program or command procedure.

When you submit a batch job, the system logs in under your user name and creates a detached process dedicated to executing the commands in that file. A batch job is also more efficient than an interactive session: it runs under a lower priority and can be run at times when the system is not overloaded with interactive users.

3.1.5 Submitting a Batch Job

To run a job in batch mode, submit your job to a batch queue (a list of batch jobs waiting to execute) by entering the DCL command SUBMIT. When you submit a job, it is directed to the default batch queue, SYS$BATCH, where it is added to the end of the queue of jobs waiting to be executed. When the jobs preceding yours are completed, your job is executed. (On a VMS system, the number of batch jobs that can execute simultaneously is specified when the batch queue is created by the system manager.)

By default, the SUBMIT command uses a file type of COM. The following command enters JOB1.COM into SYS$BATCH:

```
$ SUBMIT JOB1
Job JOB1 (queue SYS$BATCH, entry 651, started on SYS$BATCH)
```

The system displays the name of the job, the queue containing the job, and the entry number assigned to the job. You receive the DCL prompt once your job is submitted to the batch queue. If you need to reference your batch job in any DCL commands (DELETE/ENTRY, for example), do so by using the job entry number. (You can obtain the job entry number by using the SHOW ENTRY command.) Note that if

multiple procedures are submitted in a batch job, the batch job terminates when any procedure exits with an error or fatal error status.

Your batch job does not necessarily have to start running at the time you submit it to the batch queue. To specify a different time, enter the SUBMIT/AFTER command. In the following example, the job is submitted after 11:30 p.m.:

```
$ SUBMIT/AFTER=23:30 JOB1.COM
```

3.1.6 Batch Job Output

By default, accumulated output from a batch job is written to a log file once each minute. (To specify a different time interval, include the SET OUTPUT_RATE command in your command procedure.) If you attempt to use the EDT editor to read the log file while the system is writing to it, you receive a message indicating that the file is locked by another user. Wait a few seconds and try again. The EVE editor, however, allows you to read the batch job's log file. By specifying EDIT/TPU /READ_ONLY and the name of the log file, you can use EVE commands to move around the log file and ensure that any changes you make to the file are not saved. If you omit the /READ_ONLY qualifier and modify the log file in any way, the batch job terminates.

Because your batch job is a process that logs in under your user name and executes your login command procedure, the output from a batch job includes the contents of your login command procedure. The output also includes everything written to the batch job log file (command procedure output, error messages, and so on) and the full logout message. To prevent your login command procedure from being written to the batch log file, add the following command to the beginning of your login command procedure:

```
$ IF F$MODE() .EQS. "BATCH" THEN SET NOVERIFY
```

By default, the log file name is the name under which you submitted the job. Also by default, the log file has a file type of LOG and assumes the device and directory specified by your login defaults. To specify a different log file name when you submit the job, use the /LOG_NAME qualifier to the SUBMIT command.

When the batch job completes, the log file is queued to the default system printer (SYS$PRINT), printed, and deleted. To save the log file after printing it, use the /KEEP qualifier to the SUBMIT command. To save the log file without printing it, use the /NOPRINT qualifier to the SUBMIT command.

3.1.7 Restarting Batch Jobs

If the system fails while your batch job is executing, your job does not complete. When the system recovers and the queue is restarted, your job is aborted, and the next job in the queue is executed. However, by specifying the /RESTART qualifier when you submit a batch job, you indicate that the system should reexecute your job if the system crashes before the job is finished.

By default, a batch job is reexecuted beginning with the first line. See Section 6.10 for more information about symbols you can add to your command procedures to specify a different restarting point.

Chapter 4

Using Logical Names

When you define a logical name, you equate one character string to an *equivalence name,* which is usually a full or partial file specification, another logical name, or any other character string. Once you have equated a logical name to one or more equivalence names, you can use the logical name to refer to those equivalence names. For example, you might assign a logical name to your default disk and directory. Logical names serve two main functions:

- Shorthand and readability—You can define commonly used files, directories, and devices with short, meaningful logical names. Such names are easier to remember and type than the full file specifications. Names that you use frequently can be defined in your login command procedure. Names that most users on your system use frequently can be defined by a system manager in the site-specific system startup command procedure.
- File independence—You can use logical names to keep your programs and command procedures independent of physical file specifications. For example, if a command procedure references the logical name ACCOUNTS, you can equate ACCOUNTS to any file on any disk before executing the command procedure.

Logical names can be defined by you or by the system. Logical names and their definitions are kept in tables called *logical name tables.* The system provides the following logical name tables:

- Your process table
- The job table for your process
- Your group table
- The system table

When you enter a logical name as part of a command line, the system translates the logical name. It does this by searching the logical name tables in a certain order. Information about existing logical name tables and the order in which they are searched is stored in two *logical name directory tables.*

With DCL, you can also apply special attributes to logical names and define the order in which logical names tables are searched.

4.1 Creating Logical Names

You can create your own logical names with either the ASSIGN or the DEFINE command. This section uses the DEFINE command to create logical names. (Note that the syntax for the ASSIGN command differs from the syntax for the DEFINE command. For information on using the ASSIGN command, see the Reference Section.)

The syntax for defining a logical name is as follows:

DEFINE logical-name equivalence-name[,...]

The following example associates the logical name ACCOUNTS with the equivalence name DISK1:[JONES.ACCOUNTS]:

```
$ DEFINE ACCOUNTS DISK1:[JONES.ACCOUNTS]
```

Now you can use ACCOUNTS to refer to the directory DISK1:[JONES.ACCOUNTS].

Observe the following rules when creating a logical name with the DEFINE command:

- A logical name and its equivalence name can each have a maximum of 255 characters. A logical name can contain alphanumeric characters, as well as the underscore (_), dollar sign ($), and hyphen (-).
- The equivalence name must include the punctuation marks (colons, brackets, periods) that would be required if it were part of a file specification. For example, a device name is terminated by a colon, a directory specification is enclosed in square brackets, and a file type is preceded by a period.
- You can optionally terminate a logical name with a colon. If you do this, the ASSIGN command removes the colon before placing the logical name in a logical name table. The DEFINE command does not remove the colon before placing the name in a logical name table.

  In general, you should not specify a colon at the end of a logical name when you are creating it. However, if you do so and want to save the colon as part of the logical name, use the DEFINE command. (Note that when you delete a logical name ending with a colon, you need to specify two colons because the DEASSIGN command, like the ASSIGN command, removes one colon before it searches the logical name table for a match.)

If the logical name is part of a file specification, the logical name must be the leftmost component of the file specification and must be separated from the rest of the file specification by a colon. When you use a logical name to represent a complete file specification, the terminating colon is not needed. The following examples all display the file DISK1:[SALES_STAFF]PAYROLL.DAT:

```
$ DEFINE PAY DISK1:[SALES_STAFF]PAYROLL.DAT
$ TYPE PAY

$ DEFINE PAY_FILE DISK1:[SALES_STAFF]PAYROLL
$ TYPE PAY_FILE:.DAT

$ DEFINE PAY_DIR DISK1:[SALES_STAFF]
$ TYPE PAY_DIR:PAYROLL.DAT

$ DEFINE PAY_DISK DISK1:
$ TYPE PAY_DISK:[SALES_STAFF]PAYROLL.DAT
```

Note that if you combine a logical name with only an explicitly stated file type or version number, you must include the period or semicolon, respectively. For example, if PAY is equivalent to DUA1:[SALES_STAFF]PAYROLL.DAT, PAY:2 is an invalid file specification. (PAY:;2 is valid.) Defaults for the current directory, the file type (depending on the function being performed), and the version number are applied as usual after translation.

By default, the DEFINE command places logical names in your process logical name table, where the logical name is available only to your process and subprocesses. Section 4.2 describes logical name tables.

You can equate more than one logical name with an equivalence name. For example, you can equate the logical names $TERMINAL and CONSOLE to the physical name of a terminal so that both logical names translate to the same device. (If you equate a logical name to more than one equivalence string in a single command, you create a search list for the system to use to translate the names. See Section 4.7 for information about search list translation.)

If you equate a logical name to one equivalence string and then equate the same logical name to another equivalence string, the second definition supersedes the first. You can, however, equate the same logical name to different equivalence strings if the logical name definitions are in different tables (described in Section 4.2). You can equate the same logical name to different equivalence strings in the same table if they are defined in different access modes (described in Section 4.5).

If you cannot access a file and the command you are specifying and the file specification seem in order, check the left-hand component of the file specification (with SHOW LOGICAL) to be sure that it is not defined as a logical name.

4.1.1 Displaying Logical Names

Display the definition of a logical name with either the SHOW LOGICAL or SHOW TRANSLATION command.

When you enter the SHOW LOGICAL command, the system searches the process, job, group, and system logical name tables (in that order) for the specified logical name. In the following example, the system found the logical name ACCOUNTS in both the process and job logical name tables:

```
$ SHOW LOGICAL ACCOUNTS
  "ACCOUNTS" = "DISK1:[ACCOUNTS]" (LNM$PROCESS_TABLE)
  "ACCOUNTS" = "DISK1:[ACCOUNTS]" (LNM$JOB_80891AE0)
```

Sometimes the definition of a logical name may include another logical name. The SHOW LOGICAL command continues to search the logical name tables until all levels of logical names in a definition have been found. This is referred to as *iterative translation.*

When iterative translation is performed, the SHOW LOGICAL command displays multiple lines. Each line has a number that shows the level of translation. For example:

```
$ SHOW LOGICAL MYDISK
  "MYDISK" = "DISK2" (LNM$PROCESS_TABLE)
1 "DISK2" = "$11$DUA4:" (LNM$SYSTEM_TABLE)
```

Level numbers are zero based; that is, 0 is the first level, 1 is the second, and so on. In the previous example, two translations were performed. The number 1 indicates the second level of translation. See Section 4.4.1 for more information about iterative translation.

Unless you have redefined the search order, you can display the contents of the process, job, group, and system logical name tables by entering the SHOW LOGICAL command without qualifiers or parameters. The following command displays the logical names and their definitions in all four tables:

```
$ SHOW LOGICAL
```

When you enter the SHOW TRANSLATION command, the system searches the process, job, group, and system logical name tables for the specified logical name. It displays the first definition it finds as well as the table in which it was found. In the following example, you see that the logical name ACCOUNTS is translated as DISK1:[ACCOUNTS] and exists in the process logical name table (LNM$PROCESS_TABLE):

```
$ SHOW TRANSLATION ACCOUNTS
 "ACCOUNTS" = "DISK1:[ACCOUNTS]"  (LNM$PROCESS_TABLE)
```

Some commands and lexical functions do not translate logical names iteratively. The SHOW TRANSLATION command, for example, provides only the immediate equivalence name, as shown in the following example:

```
$ DEFINE SALES_DISK WORKDISK:
$ DEFINE SALES SALES_DISK
$ SHOW TRANSLATION SALES
SALES = "SALES_DISK" (LNM$PROCESS_TABLE)
```

Although the SHOW LOGICAL and SHOW TRANSLATION commands both translate logical names, certain circumstances argue for the use of one command over the other, as follows:

- To ensure that all levels of logical name translation are performed, specify the SHOW LOGICAL command. If you are certain that a logical name does not require iterative translation, specify the SHOW TRANSLATION command.
- Because the SHOW TRANSLATION command is a built-in command, you can interrupt an image (with CTRL/Y, for example) and enter SHOW TRANSLATION without causing the interrupted image to exit. (Built-in commands are described in Section 1.2.) If you interrupt an image with the SHOW LOGICAL command, your interrupted image is forced to exit.

4.1.2 Deleting Logical Names

To delete a logical name defined interactively, use the DEASSIGN command. For example:

```
$ DEFINE STAFF [JONES.STAFF]
.
.
.
$ DEASSIGN STAFF
```

Logical names in your process and job tables are automatically deleted when your process terminates. However, by specifying the /USER_MODE qualifier to the DEFINE command, you can place a logical name in the process logical name table and execute one command image before the logical name is deleted.

4.2 Logical Name Tables

The system stores logical names and their equivalence strings in four logical name tables called process, job, system, and group. Some logical name tables are available only to your process; these tables are called *process-private*. Other tables are *shareable*; that is, they are available to other users on the system.

Identical logical names can exist in more than one table. The logical name that is used depends on the order in which the logical name tables are searched. For example, when the system attempts to translate a logical name in order to identify the location of a file, it uses the logical name LNM$FILE_DEV to provide the list of tables in which to look for the name. The order in which the tables are listed is also the order in which they are searched. The precedence order defined by LNM$FILE_DEV is (1) process table, (2) job table, (3) group table, and (4) system table. Therefore, if a logical name exists in both the process and the group logical name

tables, the logical name within the process table is used. See Section 4.3.2 for more information about LNM$FILE_DEV.

Within each table, the system defines some logical names for you. Each table and its system-defined logical names are described in the following sections.

4.2.1 The Process Table

Your process logical name table, named LNM$PROCESS_TABLE, contains logical names that are available only to your process and any subsequent subprocesses. Use the logical name LNM$PROCESS to refer to the process table.

Process logical names are recognized by the process they were created in and by any subsequent subprocesses. However, process logical names are not recognized by any parent process.

To display the logical names in your process table, use the following command:

```
$ SHOW LOGICAL/PROCESS
```

You can also specify the SHOW LOGICAL/TABLE=table_name command to display the contents of any logical name table.

By default, the DEFINE and DEASSIGN commands place names in and delete names from your process table.

Every process on the system has a process logical name table. When you log in, the system creates logical names for your process and places them in your process table. These names are listed in Table 4–1.

Table 4–1: Default Process Logical Names

Logical Name	Description
SYS$COMMAND	The initial file (usually your terminal) from which DCL reads input. (A file from which DCL reads input is called an *input stream*.) The command interpreter uses SYS$COMMAND to "remember" the original input stream.
SYS$DISK	Default device established at login or changed by the SET DEFAULT command.
SYS$ERROR	The default device or file to which DCL writes error messages generated by warnings, errors, and severe errors.
SYS$INPUT	The default file from which DCL reads input.
SYS$NET	The source process that invokes a target process in DECnet–VAX task-to-task communication. When opened by the target process, SYS$NET represents the logical link over which that process can exchange data with its partner. SYS$NET is defined only during task-to-task communication.

Table 4–1 (Cont.): Default Process Logical Names

Logical Name	Description
SYS$OUTPUT	The default file (usually your terminal) to which DCL writes output. (A file to which DCL writes output is called an *output stream*.)
TT	Default device name for terminals.

Note that the logical names SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND refer to files that remain open for the life of the process. They are referred to as *process-permanent files*. For more information on process-permanent files, see Section 4.9.1.

4.2.2 The Job Table

Your job logical name table contains logical names that are available to all processes in your job tree, no matter what process or subprocess you are currently in. Your job table is named LNM$JOB_xxx, where xxx is the Job Information Block address (defined by the system) for your job tree. Use the logical name LNM$JOB to refer to your job table.

When you log in, the system creates certain logical names and places them in the job logical name table. These names are listed in Table 4–2. In addition, the logical names created for mounted disks and tapes and temporary mailboxes are also placed in the job logical name table.

Table 4–2: Default Job Logical Names

Logical Name	Description
SYS$LOGIN	Your default device and directory when you log in.
SYS$LOGIN_DEVICE	Your default device when you log in.
SYS$REM_ID	For jobs initiated through a DECnet network connection, the identification of the process on the remote node from which the job was originated. On VMS operating systems, if proxy logins are enabled, this identification is the process's user name, or, if proxy logins are not enabled, this is the process identification number (PID). (Proxy logins to proxy accounts allow users to access files across the network without specifying an access control string.)
SYS$REM_NODE	For jobs initiated through a DECnet network connection, the name of the remote node from which the job was originated.
SYS$SCRATCH	Default device and directory to which temporary files are written.

There is one job table for each job tree in the system. All job tables are shareable so that all users may access them. However, to access a job logical name table other than your own, you must redefine LNM$JOB in your process directory logical name table. For more information about LNM$JOB, see Section 4.3.

4.2.3 The Group Table

The group logical name table contains logical names that are available to all users with the same user identification code (UIC) group number. The group table is named LNM$GROUP_xxx, where xxx represents your UIC group number. Use the logical name LNM$GROUP to refer to your group table. Every group on the system has a corresponding group logical name table.

To create or delete a name in your group table, you need GRPNAM, GRPPRV, or SYSPRV privilege. See the *VMS System Manager's Manual* for a description of user privileges.

4.2.4 The System Table

The system logical name table contains logical names that are available to all users on the system. The system table is named LNM$SYSTEM_TABLE; use the logical name LNM$SYSTEM to refer to it. To create or delete a name in the system table, you must have a UIC group number between 0 and 10, or SYSNAM or SYSPRV privilege.

There is only one system logical name table for the system. It contains the names shown in Table 4–3.

Table 4–3: Default System Logical Names

Logical Name	Description
DBG$INPUT	Default input stream for the VMS Debugger; equated to SYS$INPUT
DBG$OUTPUT	Default output stream for the VMS Debugger; equated to SYS$OUTPUT
SYS$COMMON	Device and directory name for the common part of SYS$SYSROOT
SYS$ERRORLOG	Device and directory name of error log data files
SYS$EXAMPLES	Device and directory name of system examples
SYS$HELP	Device and directory name of system HELP files
SYS$INSTRUCTION	Device and directory name of system instruction data files
SYS$LIBRARY	Device and directory name of system libraries
SYS$LOADABLE_IMAGES	Device and directory of operating system executive loadable images, device drivers, and other executive loaded code
SYS$MAINTENANCE	Device and directory name of system maintenance files
SYS$MANAGER	Device and directory name of system manager files
SYS$MESSAGE	Device and directory name of system message files
SYS$NODE	Network node name for the local system if DECnet–VAX is active on the system
SYS$SHARE	Device and directory name of system shareable images

Table 4–3 (Cont.): Default System Logical Names

Logical Name	Description
SYS$SPECIFIC	Device and directory name for node-specific part of SYS$SYSDEVICE
SYS$STARTUP	Device and directory name of system startup files
SYS$SYSDEVICE	VMS system disk containing system directories
SYS$SYSROOT	Device and root directory for system directories
SYS$SYSTEM	Device and directory of operating system programs and procedures
SYS$TEST	Device and directory name of User Environment Test Package (UETP) files
SYS$UPDATE	Device and directory name of system update files

4.3 Directory Logical Name Tables

The system provides the following two directory tables to catalog your logical name tables:

- LNM$PROCESS_DIRECTORY catalogs your process tables (LNM$PROCESS and LNM$JOB).
- LNM$SYSTEM_DIRECTORY catalogs your shareable tables (LNM$GROUP and LNM$SYSTEM).

Both of these directories contain logical names that translate iteratively to table names. The name of a logical name table must be recorded in one of these directory tables in order for the system to find it.

You can see the relationship of directory tables to logical name tables with the SHOW LOGICAL/STRUCTURE command, as shown in the following example:

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$GROUP_000360)
    (LNM$JOB_806E98E0)
    (LNM$SYSTEM_TABLE)
```

4.3.1 The Process Directory Table

Each process on the system has its own process directory logical name table. When you log in, the VMS operating system places certain logical names in your process directory table. These names are listed in Table 4–4.

Table 4–4: Default Process Directory Logical Names

Logical Name	Description
LNM$GROUP	A logical name that is defined as LNM$GROUP_xxx, where xxx represents your group number. LNM$GROUP_xxx is the logical name table used by your UIC group. (The table LNM$GROUP_xxx is cataloged in the system directory table.) Therefore, LNM$GROUP is a logical name that translates iteratively to the name of your group logical name table.
LNM$JOB	A logical name that is defined as LNM$JOB_xxx, where xxx represents a number unique to your job tree. LNM$JOB_xxx is the logical name table used by your job. (The table LNM$JOB_xxx is cataloged in the system directory table.) Therefore, LNM$JOB is a logical name that translates iteratively to the name of your job logical name table.
LNM$PROCESS	A logical name that translates iteratively to LNM$PROCESS_TABLE, which is the name of your process logical name table.
LNM$PROCESS_DIRECTORY	The name of your process directory logical name table.
LNM$PROCESS_TABLE	The name of your process logical name table.

4.3.2 The System Directory Table

There is one system directory logical name table. The VMS operating system places certain logical names in the system directory table. These names are listed in Table 4–5.

Table 4–5: Default System Directory Logical Names

Logical Name	Description
LNM$DCL_LOGICAL	A logical name that is defined as LNM$FILE_DEV. This logical name iteratively translates into the list of logical name tables searched and displayed by the SHOW LOGICAL and SHOW TRANSLATION commands and the F$TRNLNM lexical function. By default, these commands search and display the process, job, group, and system logical name tables, in that order.
LNM$DIRECTORIES	A logical name that is defined as LNM$PROCESS_DIRECTORY and LNM$SYSTEM_DIRECTORY.

Table 4-5 (Cont.): Default System Directory Logical Names

Logical Name	Description
LNM$FILE_DEV	A logical name that is defined as the list of logical name tables searched by the system when processing a file specification. By default, it is defined as LNM$PROCESS, LNM$JOB, LNM$GROUP, and LNM$SYSTEM. This means that the process, job, group, and system logical name tables are searched, in that order.
LNM$GROUP_xxx	The name of a group logical name table, where xxx is a group number. There is an LNM$GROUP_xxx logical name table for each group in the system.
LNM$JOB_xxx	The name of a job logical name table, where xxx is a number unique to this job tree. There is an LNM$JOB_xxx logical name table for each job in the system.
LNM$SYSTEM	A logical name that translates iteratively to LNM$SYSTEM_TABLE, which is the name of the system logical name table.
LNN$SYSTEM_DIRECTORY	The name of the system directory logical name table.
LNM$SYSTEM_TABLE	The name of the system logical name table.

Generally, you do not need to change the default logical name table definitions set up in the directory tables, LNM$PROCESS_DIRECTORY and LNM$SYSTEM_DIRECTORY. Two reasons for changing the entries in the directory tables are (1) to create another logical name table, and (2) to change the search order for file specification logical names by redefining LNM$FILE_DEV. See Section 4.6 for information about creating your own logical name table and changing the order in which the system searches the logical name tables.

Multiple tables with the same name may exist. For example, there may exist both a process-private and a shareable table called MY_TABLE. The process-private version always takes precedence over the shareable table in all logical name table processing. When a logical name, such as LNM$FILE_DEV, is used as a table name, the logical name is iteratively translated until a list of table names is formed. During this iterative translation, each name is first translated in the process directory. If this translation fails, it is then translated in the system directory. This order of precedence cannot be changed. As a consequence of this ordering, a logical name placed in the process directory table for use as a table name will always take precedence over any identical name residing in the system directory.

4.4 Logical Name Translation

When the system reads a file specification or device name in a DCL command line, it examines the file specification or device name to see whether the leftmost component is a logical name. If the leftmost component ends with a colon, space, comma, or a line terminator (for example, RETURN), the system attempts to translate it as a logical name. If the leftmost component ends with any other character, the system does not attempt to translate it as a logical name.

After you enter the command shown in the following example, the system checks to see whether PUP is a logical name because PUP is the leftmost component of the file specification. Because the leftmost component is terminated with RETURN, the system attempts to translate PUP.

```
$ TYPE PUP
```

After you enter the command shown in the next example, the system checks whether DISK is a logical name. The system attempts to translate DISK because it is the leftmost component and ends with a colon. (The system does not check PUP.)

```
$ TYPE DISK:PUP
```

In the third example, the system does not try to translate [DRYSDALE]PUP because the leftmost component ends with a square bracket (]):

```
$ TYPE [DRYSDALE]PUP
```

By default, when the system translates logical names in file specifications, it searches the process, job, group, and system tables in that order, and uses the first match it finds.

4.4.1 Iterative Translation

Logical name translation can be iterative. This means that after the system translates a logical name, it repeats the translation process for any logical names it finds contained within the first logical name. For example:

```
$ DEFINE DISK DUA1:
$ DEFINE MEMO DISK:[JEFF.MEMOS]COMPLAINT.TXT
```

In this example, the first DEFINE command equates the logical name DISK to the device name DUA1. The second DEFINE command equates the logical name MEMO to the file specification DISK:[JEFF.MEMOS]COMPLAINT.TXT. When the system translates the logical name MEMO, it finds the equivalence name DISK:[JEFF.MEMOS]COMPLAINT.TXT. It then checks to see whether the leftmost component in this file specification ends in a colon, a space, a comma, or an end-of-line terminator. It finds a colon after DISK. The system translates that logical name also. The final translation of the file specification is as follows:

```
DUA1:[JEFF.MEMOS]COMPLAINT.TXT
```

The system limits the number of levels to which it performs logical name translation. The number of levels varies among system facilities, but it is at least nine. If you define more than the system-determined number of levels, or if you create a circular definition, an error occurs when the logical name is used.

4.4.2 Modifying Logical Name Translation

When you create a logical name, you can specify translation attributes that modify how the system interprets the equivalence name. Use the /TRANSLATION_ATTRIBUTES qualifier to the DEFINE command. (This is a positional qualifier: depending on where you place it on a command line, it can apply translation attributes to all equivalence names or only to certain ones.) Two translation attributes can be specified as values to the /TRANSLATION_ATTRIBUTES qualifier: CONCEALED and TERMINAL.

The CONCEALED attribute causes the logical name of a device, rather than the physical name, to be displayed in system messages (except for the SHOW LOGICAL display). The CONCEALED attribute is usually specified when defining logical names for devices. Using concealed devices allows you to write programs and command procedures and perform other operations without being concerned about which physical device actually holds the disk or tape. It also lets you use names that are more meaningful than the physical device names.

The following example shows how to create a concealed device name:

```
$ DEFINE/TRANSLATION_ATTRIBUTES=CONCEALED DISK DJA3:
$ SHOW DEFAULT
  DISK:[SAM.PUP]
$ SHOW LOGICAL DISK
  "DISK" = "DJA3" (LNM$PROCESS_TABLE)
```

The logical name DISK represents the physical device DJA3. Thus, the SHOW DEFAULT command displays the logical name DISK rather than the actual physical device name, DJA3. The SHOW LOGICAL command reveals the translation of DISK.

The TERMINAL attribute prevents iterative translation of a logical name. That is, the equivalence name is not examined to see if it is also a logical name. The translation is "terminal" (final, or completed) after the first translation.

4.4.3 System Defaults During Logical Name Translation

When the system translates a logical name, it fills in any missing fields in a file specification. It fills them in with the current default device, directory, and version number. When you use a logical name to specify the input file for a command, the command uses the logical name to assign a file specification to the output file as well.

If the equivalence name contains a file name and file type, the output file is given the same file name and file type. If the equivalence name does not contain a file type, a default file type is supplied. The file type supplied depends on the command you are using.

4.5 Logical Name Access Modes

The four access modes in the VMS operating system are as follows:

- User-mode (the outermost and least privileged mode)
- Supervisor-mode
- Executive-mode
- Kernel-mode (the innermost and most privileged mode)

When you create a logical name with DCL commands, it has an access mode of user, supervisor, or executive. By default, logical names are created in supervisor mode; you must have SYSNAM privilege to create an executive mode logical name. To see the access mode for a logical name, use the SHOW LOGICAL/FULL command, as follows:

```
$ SHOW LOGICAL/FULL PROJECT
  "PROJECT" [super] = "DISK1:[JONES]" (LNM$PROCESS_TABLE)
```

This shows that the logical name PROJECT was created in supervisor mode.

You can equate the same logical name to different equivalence strings in the same logical name table by specifying different access modes for each definition. The following example equates the logical name ACCOUNTS to two different equivalence names in the process logical name table—one in supervisor-mode and one in executive-mode:

```
$ DEFINE ACCOUNTS DISK1:[ACCOUNTS]CURRENT.DAT
$ DEFINE/EXECUTIVE_MODE ACCOUNTS DISK1:[JANE.ACCOUNTS]OBSOLETE.DAT
```

Logical names created in user mode are temporary. Define a logical name in user mode when you want to define it only for the execution of the next image. In the following example, the logical name ADDRESSES is automatically deleted after the execution of the program PAYABLE:

```
$ DEFINE/USER_MODE ADDRESSES DISK1:[SAM.ACCOUNTS]OVERDUE.LIS
$ RUN PAYABLE
```

In looking up logical names, all privileged images and utilities, such as LOGINOUT and MAIL, bypass the user- and supervisor-mode portions of the system logical name table. Therefore, DIGITAL recommends that logical names for important system components (public disks and directories, for example) be defined in executive mode, using the DCL command DEFINE/SYSTEM/EXECUTIVE. (Only the operating system and privileged programs can create logical names in kernel-mode.) This operation requires either the SYSPRV or SYSNAM privilege.

4.6 Creating a Logical Name Table

The CREATE/NAME_TABLE command creates a logical name table and catalogs it in one of the directory logical name tables. (Logical names that identify logical name tables or that translate iteratively to logical name tables must always be entered into one of the directory logical name tables.) To create a logical name table that is private to your process, create the table in LNM$PROCESS_DIRECTORY (the default). If you want the table to be shareable, specify /PARENT_TABLE=LNM$SYSTEM_DIRECTORY with the CREATE/NAME_TABLE command. Creating shareable name tables requires SYSPRV privilege or ENABLE access to the parent table.

The following example creates a process-private logical name table named TAX, places the definition for the logical name CREDIT in the table, and verifies the table's creation. (You must specify the /TABLE qualifier with the SHOW LOGICAL command to display a logical name in any table other than LNM$SYSTEM or LNM$PROCESS.)

```
$ CREATE/NAME_TABLE TAX
$ DEFINE/TABLE=TAX CREDIT [ACCOUNTS.CURRENT]CREDIT.DAT
$ SHOW LOGICAL/TABLE=TAX CREDIT

 "CREDIT" = "[ACCOUNTS.CURRENT]CREDIT.DAT"  (TAX)
```

To make the system search a user-created logical name table automatically when processing file specifications, you must create a process-private version of the default search list (LNM$FILE_DEV) in LNM$PROCESS_DIRECTORY. To add the created table's name to the default search list, you can define LNM$FILE_DEV as follows:

```
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$FILE_DEV -
_$ TAX,LNM$PROCESS,LNM$JOB,LNM$GROUP,LNM$SYSTEM
```

Placing the table's name first specifies that the system search that table first, and so on in the order of specification.

To delete a logical name table, specify the table that contains it (the system or process directory logical name table) and the name of the table. Deleting a shareable logical name table requires DELETE access to the table or SYSPRV privilege. For example, to delete the logical name table TAX of the preceding example, specify the following command line:

```
$ DEASSIGN/TABLE=LMN$PROCESS_DIRECTORY TAX
```

Note that all logical names in descendant tables (and the descendant tables themselves) are deleted when a parent logical name table is deleted.

4.7 Search Lists

A search list is a logical name that has more than one equivalence name. You can use a search list in any place you can use a logical name. For example:

```
$ DEFINE GETTYSBURG [JONES.HISTORY],[JONES.WORKFILES]
$ SHOW LOGICAL GETTYSBURG

  "GETTYSBURG" = "[JONES.HISTORY]" (LNM$PROCESS_TABLE)
      = "[JONES.WORKFILES]"
```

The logical name GETTYSBURG is a search list because it has more than one equivalence name.

When you use a logical name that is a search list, the system translates the logical name until it finds a match. The order in which you specify the equivalence strings determines the order in which the system translates the names. It uses each equivalence name listed in the definition until a match is found.

A search list is not a wildcard. It is a list of places to look. Once a file is found, the search is ended. For example:

```
$ TYPE GETTYSBURG:SPEECH.TXT

DISK1:[JONES.HISTORY]SPEECH.TXT;2

Fourscore and seven years ago, our fathers brought forth on
this continent a new nation, conceived in liberty, and
dedicated to the proposition that all men are created equal.
    .
    .
    .
```

In the previous example, the TYPE command searches the equivalence names [JONES.HISTORY] and [JONES.WORKFILES] in the order they were listed when GETTYSBURG was defined. Once it finds a file named SPEECH.TXT, the search is halted and the file is displayed.

You can use a search list with a command that accepts wildcards. When you use wildcards, the system forms file specifications using each equivalence name in the search list. The command operates on each file specification that identifies an existing file.

For example, if you specify the DIRECTORY command with a wildcard character in the version field, it finds all versions of SPEECH.TXT in the search list defined by GETTYSBURG, as shown in the following example:

```
$ DIRECTORY GETTYSBURG:SPEECH.TXT;*

Directory DISK1:[JONES.HISTORY]

SPEECH.TXT;2     SPEECH.TXT;1

Total of 2 files.

Directory DISK1:[JONES.WORKFILES]
```

```
SPEECH.TXT;1

Total of 1 file.

Grand total of 2 directories, 3 files.
```

The DIRECTORY command searches the equivalence names [JONES.HISTORY] and [JONES.WORKFILES] in the order they were listed when GETTYSBURG was defined. It finds a file named SPEECH.TXT in each directory. If SPEECH.TXT exists in only one of the directories, only one directory listing is displayed. If SPEECH.TXT does not exist in either directory, an error message is displayed indicating that the file was not found.

When you use a search list with a command that does not accept wildcards in a file specification, the system forms a file specification using each equivalence name in the search list until a file specification for an existing file is found. The command affects only the first file found. For example:

```
$ DEFINE DECEMBER DISK1:[FRED],WORK2:[BARNEY]
$ EDIT/EDT DECEMBER:QUOTAS.TXT
```

First, the system forms the file specification DISK1:[FRED]QUOTAS.TXT and searches for that file. If QUOTAS.TXT is found in DISK1:[FRED], it is opened for editing. No other files are subsequently opened. If QUOTAS.TXT is not found in DISK1:[FRED], the system searches for it in WORK2:[BARNEY]. If QUOTAS.TXT is found there, it is opened. If it is not found, an error message is displayed. The system displays an error message only after it checks all equivalence names in a search list. Then the system reports an error only on the last file it attempted to find.

The RUN command is an exception. When the RUN command is followed by a search list, the system forms file specifications as described previously. However, the system then checks to see whether any of the files in the list are installed images. It runs the first file in the search list that is an installed image. Then the RUN command terminates.

If none of the file specifications are installed images, the system repeats the process of forming file specifications. This time it looks for each file specification on the disk. It runs the first file it finds there. An error message is displayed if none of the specified files is found in either the known file list or on the disk.

4.8 Logical Node Names

A logical node name is a special type of logical name that can be used in place of a network node name or in place of a node name and an access control string. For example:

```
$ DEFINE BOS "BOSTON""ADAMS JOHN""::"
```

The logical name BOS is equated to the node name BOSTON and an access control string, where ADAMS is the user name and JOHN is the password. Use the logical name BOS to avoid typing (and displaying) your user name and password on the terminal screen.

NOTE: Do not place a DEFINE command that includes a password in a file (your login command procedure, for example). If others read the file, they will see the password.

4.9 System-Created Logical Names

The system creates a number of logical names for you when you start the system and log in. By default, DCL creates and assigns logical names to four process-permanent files. When you redefine these logical names, only your process is affected. The system defines other logical names that you can reassign only with special privileges.

4.9.1 Process-Permanent Logical Names

Process-permanent logical names are created by DCL when you log in and remain defined for the life of your process. You cannot deassign these logical names. You can redefine them (by specifying the same name in a DEFINE command), but if the redefined name is later deassigned, the process-permanent name is reestablished. These process-permanent logical names, as follows, are available to each user of the system at the process level:

- SYS$INPUT—Logical name that refers to the default input device or file
- SYS$OUTPUT—Logical name that refers to the default output device or file
- SYS$ERROR—Logical name that refers to the default device or file to which the system writes messages
- SYS$COMMAND—Logical name that refers to the value of SYS$INPUT when you log in

Table 4–6 shows what these logical names are equated to by default.

Table 4–6: Equivalence Names for Process-Permanent Logical Names

Logical Name	Interactive Mode	Batch Mode	Command Procedure
SYS$COMMAND	Terminal[1]	Disk[2]	Terminal
SYS$INPUT	Terminal	Disk	Disk
SYS$ERROR	Terminal	Log file[3]	Terminal
SYS$OUTPUT	Terminal	Log file	Terminal

[1]Device name of your terminal

[2]Device name of the initial default device

[3]Batch job log file

The following sections describe how to use process-permanent logical names as file specifications.

4.9.1.1 Redefining SYS$INPUT

You can redefine SYS$INPUT so that a command procedure reads input from the terminal or another file. For example, to edit a file from a command procedure, include the following lines in the command procedure:

```
$ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND
$ EDIT/TPU MYFILE.DAT
    .
    .
    .
```

In the previous example, SYS$INPUT is redefined as SYS$COMMAND so that the editor obtains input from the terminal rather than from the command procedure file (the default). SYS$COMMAND refers to the terminal, the initial input stream when you logged in. The /USER_MODE qualifier tells the command procedure that SYS$INPUT is redefined only for the duration of the next image. In this example, the next image is the editor. When the editor is finished, SYS$INPUT resumes its default value; in this case, the default value is the command procedure file.

Note that if you redefine SYS$INPUT, DCL ignores your definition. DCL always obtains input from the default input stream. However, images, such as command procedures, can use your definition for SYS$INPUT.

4.9.1.2 Redefining SYS$OUTPUT

You can redefine SYS$OUTPUT to redirect output from your default device to another file. When you redefine SYS$OUTPUT, the system opens a file with the name you specify in the logical name assignment. When you define SYS$OUTPUT, all subsequent output is directed to the new file.

In the following example, SYS$OUTPUT is defined as MYFILE.LIS before the SHOW DEVICES command is entered. The display produced by SHOW DEVICES is directed to MYFILE.LIS in your current directory rather than to your terminal. You can manipulate this data as you would any other text file.

```
$ DEFINE SYS$OUTPUT MYFILE.LIS
$ SHOW DEVICES
```

Remember to deassign SYS$OUTPUT, or output will continue to be written to the file you specify. Note that you can redefine SYS$OUTPUT in user mode (with DEFINE /USER_MODE) to redirect output from an image. This definition is in effect only until the next command image is executed. Once the command image is executed (that is, the output is captured in a file), the logical name SYS$OUTPUT resumes its default value.

When you log in, the system creates two logical names called SYS$OUTPUT. One name is created in executive mode; the other name is created in supervisor mode. You can supersede the supervisor mode logical name by redefining SYS$OUTPUT. If you deassign the supervisor mode name, the system redefines SYS$OUTPUT in supervisor mode, using the executive mode equivalence name. You cannot deassign the executive mode name.

In the following example, SYS$OUTPUT is redefined to the file TEMP.DAT. When SYS$OUTPUT is redefined, output from DCL and from images is directed to the file TEMP.DAT. The output from the SHOW LOGICAL command and from the SHOW TIME command is also sent to TEMP.DAT. When you deassign SYS$OUTPUT, the system closes the file TEMP.DAT and redefines SYS$OUTPUT to your terminal. When you enter the TYPE command, the output collected in TEMP.DAT is displayed on your terminal.

```
$ DEFINE SYS$OUTPUT TEMP.DAT
$ SHOW LOGICAL SYS$OUTPUT
$ SHOW TIME
$ DEASSIGN SYS$OUTPUT
$ TYPE TEMP.DAT
   "SYS$OUTPUT" = "DISK1:" (LNM$PROCESS_TABLE)
  31-DEC-JAN-1988 13:26:53
```

When you redefine SYS$OUTPUT to a file, the logical name contains only the device portion of the file specification, even though the output is directed to the file you specify. In the previous example, when SYS$OUTPUT was redefined, the equivalence name contained the device name DISK1, not the full file specification.

If the system cannot open the file you specify when you redefine SYS$OUTPUT, an error message is displayed.

After you redefine SYS$OUTPUT, most commands direct output to the existing version of the file. However, certain commands create a new version of the file before they write output.

4.9.1.3 Redefining SYS$ERROR

You can redefine SYS$ERROR to direct error messages to a specified file. However, if you redefine SYS$ERROR so it is different from SYS$OUTPUT (or if you redefine SYS$OUTPUT without also redefining SYS$ERROR), DCL commands send informational, warning, error, and severe error messages to both SYS$ERROR and SYS$OUTPUT. Therefore, you receive these messages twice—once in the file indicated by the definition of SYS$ERROR and once in the file indicated by SYS$OUTPUT. Success messages are sent only to the file indicated by SYS$OUTPUT.

If you redefine SYS$ERROR and then run an image that references SYS$ERROR, the image sends error messages only to the file indicated by SYS$ERROR even if SYS$ERROR is different from SYS$OUTPUT. Only DCL commands and images using standard VMS error display mechanisms send error messages to both SYS$ERROR and SYS$OUTPUT when these files are different.

4.9.1.4 Redefining SYS$COMMAND

Although you can redefine SYS$COMMAND, DCL ignores your definition. DCL always uses the default definition for your initial input stream. However, if you execute an image that references SYS$COMMAND, the image can use your new definition.

4.9.2 System-Permanent Logical Names

The following table lists the logical names automatically defined when the system starts up. These names are available to all users of the system at the system level.

Logical Name	Equivalence Name
DBG$INPUT	SYS$INPUT at the process level
DBG$OUTPUT	SYS$OUTPUT at the process level
SYS$COMMON	SYS$SYSDEVICE:[SYS*n*.SYSCOMMON.], where *n* is the root directory number of your processor
SYS$ERRORLOG	SYS$SYSROOT:[SYSERR]
SYS$EXAMPLES	SYS$SYSROOT:[SYSHLP.EXAMPLES]
SYS$HELP	SYS$SYSROOT:[SYSHLP]
SYS$INSTRUCTION	SYS$SYSROOT:[SYSCBI]
SYS$LIBRARY	SYS$SYSROOT:[SYSLIB]
SYS$LOADABLE_IMAGES	SYS$SYSROOT:[SYS$LDR]
SYS$MAINTENANCE	SYS$SYSROOT:[SYSMAINT]
SYS$MANAGER	SYS$SYSROOT:[SYSMGR]
SYS$MESSAGE	SYS$SYSROOT:[SYSMSG]
SYS$NODE	Name of your node if you are on a network
SYS$SHARE	SYS$SYSROOT:[SYSLIB]
SYS$SPECIFIC	SYS$SYSDEVICE:[SYS*n*.], where *n* is the root directory number of your processor
SYS$STARTUP	As a search list, points first to SYS$SYSROOT:[SYS$STARTUP], then to SYS$MANAGER
SYS$SYSDEVICE	System disk (usually SYS$DISK)
SYS$SYSROOT	As a search list, points first to SYS$SYSDEVICE:[SYS*n*.], where *n* is the root directory number of your processor; then to SYS$COMMON
SYS$SYSTEM	SYS$SYSROOT:[SYSEXE]
SYS$TEST	SYS$SYSROOT:[SYSTEST]
SYS$UPDATE	SYS$SYSROOT:[SYSUPD]

Chapter 5

Representing Data with Symbols

As you carry out your computing tasks with the support of DCL and VMS, you may need to store and manipulate data, such as numbers and strings of characters, through the use of symbols. Like files, symbols store data. Yet, unlike files, the symbols you create are temporary and have no means of physical storage—they exist only for the life of your computing session or for the life of a program's execution.

This chapter describes the kinds of data you can use in DCL, how you can use symbols to represent that data, and how you can combine symbols into expressions to manipulate the data that the symbols represent.

5.1 Data Storage

With the VMS operating system, the most common units in which data can be stored are the following:

- Bit—The most basic unit of storage, a bit has a value of 0 or 1.
- Byte—Equal to 8 bits, a byte can represent an unsigned integer value of 0 through 255 and a signed value of –128 through 127. Characters are stored one per byte.
- Word—Equal to 2 bytes, a word can represent an unsigned integer value of 0 through 65,353 and a signed value of –32,768 through 32,767.
- Longword—Equal to 4 bytes (32 bits), a longword can represent an unsigned integer value of 0 through 4,294,967,295 and a signed value of –2,147,483,648 through 2,147,483,647.

The first unit in any series of these units is called the *low-order* unit. In numeric values, the low-order unit is the least-significant unit in the number. For example, in a binary number composed of the series of bits 11111110, the 0 is the low-order bit.

5.2 Creating and Using Symbols

A *symbol* is a name that represents a character value (for example, "DOG"), a numeric value (for example, 17), or a logical value (for example, True). When you use a symbol in a DCL command, DCL replaces the symbol with its value before executing the command. Symbols are useful for representing data in commands and command procedures and as shortcuts for entering commands you use frequently.

For example, you may define a symbol as any of the following:

- Foreign command—Defining a symbol as a foreign command allows you to execute an image by entering only the symbol name. (The command is "foreign" because it is unknown to DCL.) In the following example, the symbol FIX is defined to execute the image NUMFIX.EXE in the [BILLS] directory on the disk ACCOUNTS:

  ```
  $ FIX == "$ACCOUNTS:[BILLS]NUMFIX"
  ```

- Command line—Defining a symbol as a command line allows you to execute the command by entering only the symbol name. In the following example, the symbol HUBBUB is defined to establish a network connection to the node HUBBUB:

  ```
  $ HUBBUB == "SET HOST HUBBUB"
  ```

  Setting a symbol equal to a command line that executes a command procedure allows you to execute the procedure by typing only the symbol name. In the following example, COUNT is defined to execute the command procedure CENSUS:

  ```
  $ COUNT == "@DISK1:[JONES.PROCEDURES]CENSUS"
  ```

  When you enter COUNT to execute CENSUS, place any parameters for CENSUS after the symbol as if you had entered @CENSUS.

- Character string—Defining a symbol as a character string allows you to insert that string in a command line by typing the symbol (with surrounding apostrophes to force substitution, as described in Section 5.5). In the following example, the symbol FILE is first defined as a complete file specification and then used in the TYPE command:

  ```
  $ FILE == "DISK1:[JONES.TAXES]CORPORATE.DAT"
  $ TYPE 'FILE'
  ```

  The string can be a directory you often access. In the following example, whenever the symbol TAXES occurs in a command line, the literal value replaces the symbol before the line is executed.

  ```
  $ TAXES == "[JONES.TAXES]"
  $ COPY 'TAXES'OVERDUE.DAT OVERDUE.TMP
  ```

Symbols can also hold variables, which are values that you calculate or that you assign as something other than a literal. For example, you might assign the value of a lexical function to a variable or read the value of a file record into a variable. As variables, symbols are most often used in command procedures (see Chapter 6).

To create a symbol, assign a value to a symbolic name using the following format:

symbol-name =[=] value

The symbol name can be 1 through 255 characters long and must begin with a letter, an underscore (_), or a dollar sign ($). In a symbol name, both lowercase and uppercase letters are treated as uppercase.

The value you assign to a symbol can be made either locally or globally available to the command interpreter:

- Local—A local symbol is available to the command level that defined it, to any command procedure executed from that level, and to lower command levels. (By convention, DCL level—command level 0—is the highest command level and command level 31 the lowest command level. Thus, when you move from command level 3 to command level 2, you are moving to the next higher command level. If you execute a command procedure interactively, the commands in the procedure are executed at command level 1. You can create a maximum of 32 command levels.)
- Global—A global symbol is available to any command level regardless of the level at which it was defined.

To create a local symbol, use a single equal sign in the assignment statement. To create a global symbol, use two equal signs. The following commands define the local symbol FILE as the character string DISK2:[BOLIVAR]PRICES.CUR and the global symbol MAX_VALUE as the number 24.

```
$ FILE = "DISK2:[BOLIVAR]PRICES.CUR"
$ MAX_VALUE == 24
```

You can omit the quotation marks around character strings in assignment statements if you precede the equal sign or signs with a colon. Symbol assignments that omit quotation marks automatically change the character string to uppercase letters and compress multiple spaces and tabs to a single space. The following example again creates the local symbol FILE, this time omitting the quotation marks because of the included colon:

```
$ FILE := ACCOUNTS:[BOLIVAR]PRICES.84
```

The result of DCL's evaluation of a symbol is either a character string or an integer value. The data type (character or integer) of a symbol is determined by the data type of the value currently assigned. The type is not permanent: if the value changes type, as in the following example, the symbol changes type. In the following example, the local symbol NUM is first assigned a character value and then converted to an integer value when used in an expression with an integer:

```
$ NUM = "12"
$ RESULT = NUM + 10
```

Local symbols take precedence over global symbols with the same name. Symbols take precedence over identical command names. This means that if you define a symbol with the same name as a DCL command, your definition overrides the command name. For example, if you define the symbol HELP as the command TYPE HELP.LST, you can no longer invoke the system's HELP facility by typing HELP.

Symbols are stored in the following tables, which are maintained by the operating system:

- Local symbol table—DCL maintains a local symbol table for your main process and for every command level that you create when you execute a command procedure, use the CALL command, or submit a batch job. A local table is deleted when its associated command level terminates. (See Chapter 3 for more information about processes, command procedures, and batch jobs.)

  In addition to the local symbols you create, a local symbol table contains eight symbols that are maintained by DCL. These symbols, named P1, P2, and so on through P8, are used for passing parameters to a command procedure. Parameters passed to a command procedure are regarded as character strings. Otherwise, P1 through P8 are defined as null character strings (""). They are stored in the local symbol table.

- Global symbol table—DCL maintains only one global symbol table for the duration of a process. In addition to the global symbols you create, the global symbol table contains the reserved global symbols described in the following table. These global symbols give you status information on your programs and command procedures as well as on system commands and utilities.

Reserved Global Symbols	Definition
$STATUS	The condition code returned by the most recently executed command. $STATUS conforms to the format of a VMS message code. User programs can set the value of the global symbol $STATUS by including a parameter value to the EXIT command. The system uses the value of $STATUS to determine which message, if any, to display and whether to continue execution at the next-higher command level. The value of the lower three bits in $STATUS is placed in the global symbol $SEVERITY.
$SEVERITY	The severity level of the condition code returned by the most recently executed command. $SEVERITY, which is equal to the lower three bits of $STATUS, can have the following values: 0 Warning 1 Success 2 Error 3 Information 4 Severe (fatal) error
$RESTART	Has the value TRUE if a batch job was restarted after it was interrupted by a system crash. Otherwise, $RESTART has the value FALSE.

5.3 Abbreviating Symbol Names

You can use abbreviated forms of symbols if you define them with the asterisk. The following example shows how to create a local symbol that can be abbreviated:

```
$ M*AIL = "MAIL"
```

Once this symbol is established, the VMS Mail Utility is invoked whenever you specify any of the following versions of the symbolic name:

```
$ M
$ MA
$ MAI
$ MAIL
```

Generally, you can use abbreviated symbol definitions in any situation that allows a symbol to be used. However, there are some restrictions as follows:

- You cannot abbreviate symbols that involve substring replacement.
- When you define a symbol that includes an asterisk, existing symbols may possibly be deleted. If an existing symbol exactly matches the new symbol at or past the asterisk, the new symbol replaces the existing symbol.
- If you define a symbol with an asterisk, you cannot define another symbol whose name partly matches the existing symbol at or past the asterisk.

5.4 DCL Commands to Use with Symbols

Table 5–1 shows the DCL commands you can use with symbols.

Table 5–1: DCL Commands to Use with Symbols

Command	Function
SHOW SYMBOL	Displays the value of the specified symbol. By default, the SHOW SYMBOL command searches the local symbol tables and then the global symbol table to locate a specified symbol name.
DELETE/SYMBOL	Deletes a symbol. By default, the DELETE/SYMBOL command searches for symbols only in the local symbol table. To delete a global symbol, use the /GLOBAL qualifier.
SET SYMBOL/SCOPE	You can mask global or local symbols at the specified command level.
INQUIRE	Reads a value from SYS$COMMAND and assigns it to a symbol.
READ	Reads a record from a file and assigns its contents to a symbol.

The SHOW SYMBOL command displays symbol values. Specify the name of the symbol to display the value of a particular local symbol. Specify the name of the symbol and /GLOBAL to display the value of a particular global symbol. Specify /ALL to display all local symbols and /ALL/GLOBAL to display all global symbols.

The DELETE/SYMBOL command deletes a symbol. You must include the /GLOBAL qualifier to delete a global symbol. In the following example, the global symbol TEMP is deleted:

```
$ DELETE/SYMBOL/GLOBAL TEMP
```

The SET SYMBOL/SCOPE=(keyword, . . .) command controls access to local and global symbols in command procedures. This allows you to treat symbols as undefined without deleting them. Symbol scoping works differently for local and global symbols.

If you specify /SCOPE=NOLOCAL, all local symbols defined in an outer procedure level are treated as undefined by the current procedure and any inner levels. Specifying LOCAL removes any symbol translation limit set by the current procedure level.

For example, if SET SYMBOL/SCOPE=NOLOCAL was specified at procedure levels 2 and 4, procedure level 2 can access only procedure level 2 local symbols. Procedure level 3 can access procedure levels 2 and 3 local symbols; procedure level 4 can access procedure level 4 local symbols and any local symbols in inner procedure levels.

Global symbols are not procedure level dependent. The global symbol scoping state (GLOBAL or NOGLOBAL) that is in effect when a new procedure level is invoked is propagated to the new procedure level. Specifying /SCOPE=NOGLOBAL makes all global symbols inaccessible for all subsequent commands until you either specify /SCOPE=GLOBAL or exit to a previous level at which global symbols were accessible.

In the following example, the SET SYMBOL command denies access to all global symbols:

```
$ SET SYMBOL/SCOPE=NOGLOBAL
```

Exiting a procedure level back to an outer procedure level causes the symbol scope-state to be restored for both local and global symbols.

The INQUIRE and READ commands are most often used within DCL command procedures and are therefore discussed in Chapter 6.

5.5 Symbol Substitution

When a command line is executed, symbols in the following positions are automatically substituted:

- On the right side of an [:]= or [:]== assignment statement
- In a lexical function
- In the brackets on the left side of an assignment statement when you are performing substring substitution or number overlays (see Section 5.6.2.4)
- In a DEPOSIT, EXAMINE, IF, or WRITE command
- At the beginning of the command line

To force substitution of a symbol that is not in one of the positions listed, enclose the symbol with apostrophes as follows:

```
$ TYPE 'B'
```

To force substitution of a symbol within a quoted character string, precede that symbol with double apostrophes and follow it with a single apostrophe as follows:

```
$ T = "TYPE ''B'"
```

When processing a command line, DCL replaces symbols with their values in the following order:

- Forced substitution—From left to right, DCL replaces all strings delimited by apostrophes (or double apostrophes for strings within quotation marks). Symbols preceded by single apostrophes are translated iteratively; symbols preceded by double apostrophes are not.

- Automatic substitution—From left to right, DCL evaluates each value in the command line, executing it if it is a command and evaluating it if it is an expression. Symbols in expressions are replaced by their assigned values; this substitution is not iterative.

The following example demonstrates the effect of the order in which DCL substitutes symbols. Assume the following symbol definitions:

```
$ PN = "PRINT/NOTIFY"
$ FILE1 = "[BOLIVAR]TEST_CASE.TXT"
$ NUM = 1
```

Given the preceding symbol definitions, the following commands print the file named [BOLIVAR]TEST_CASE.TXT:

```
$ FILE = "'FILE''NUM'"
$ PN 'FILE'
```

In the first command, forced substitution causes NUM to become 1, making FILE"NUM' become FILE1. If you enter the command SHOW SYMBOL FILE, you will see that FILE = "'FILE1'".

The second command performs two substitutions. First, 'FILE' is substituted with 'FILE1'. 'FILE1' also requires substitution because it is enclosed in single quotation marks. Automatic substitution causes FILE1 to become [BOLIVAR]TEST_CASE.TXT. This file name is then appended to the value of PN, which is PRINT/NOTIFY. The resulting string is as follows:

```
$ PRINT/NOTIFY [BOLIVAR]TEST_CASE.TXT
```

5.6 Storing and Manipulating Data with Symbols

You can use symbols to store and manipulate a variety of values. This section describes the values that can be stored in symbols. It also describes how symbols can be combined in expressions to manipulate the values the symbols contain.

5.6.1 Symbol Values

A symbol can be defined as a character string, a number, a lexical function, a logical value, or another symbol. The following sections describe these values.

5.6.1.1 Character String Values

A character string can contain any characters that can be printed. Appendix A, which includes tables of the ASCII character set and the DEC Multinational Character Set, shows those characters that you can include in a character string.

Characters fall into the following three main categories:

- Alphanumeric characters—The uppercase letters A through Z, the lowercase letters a through z, the digits 1 through 9, the dollar sign ($), the underscore (_), and the hyphen (-).
- Special characters—All other characters that can be displayed or printed: the exclamation point (!), quotation marks ("), number sign (#), and so on.
- Nonprintable characters—All characters that cannot be printed or displayed. In general, nonprintable characters are ignored for display and print purposes. However, several nonprintable characters serve control functions as follows:

Character	Function
HT	Starts printing or typing at the next horizontal tab
LF	Starts printing or typing on the next line
FF	Starts printing or typing at the top of the next page
CR	Starts printing or typing at the first space on the same line
ESC	Introduces a terminal escape sequence
SP	Inserts one space

You can define a character string by enclosing it in quotation marks. In this way, alphabetic case and spaces are preserved when the symbol assignment is made.

5.6.1.2 Numeric Values

A number can have the following values:

- Decimal—The ASCII characters 0 through 9
- Hexadecimal—The ASCII characters 0 through 9 and A through F
- Octal—The ASCII characters 0 through 7

The number you assign to a symbol must be in the range -2147483648 through 2147483647 (decimal). (An error is not reported if a number outside this range is specified or calculated, but an incorrect value results.)

At DCL command level and within command procedures, specify a number as follows:

- Positive numbers—Specify a positive number by typing the appropriate digits. The following example assigns the number 13 to the symbol DOG_COUNT:

```
$ DOG_COUNT = 13
$ SHOW SYMBOL DOG_COUNT
  DOG_COUNT = 13   Hex = 0000000D  Octal = 00000000015
```

- Negative numbers—Precede a negative number with a minus sign, as in the following example:

```
$ BALANCE = -15237
$ SHOW SYMBOL BALANCE
  BALANCE = -15237   Hex = FFFFC47B  Octal = 37777742173
```

- Radix—Specify a number in a radix other than decimal by preceding the number (but not the minus sign) with %X for hexadecimal numbers and %0 for octal numbers. For example:

```
$ DOG_COUNT = %XD
$ SHOW SYMBOL DOG_COUNT
  DOG_COUNT = 13   Hex = 0000000D  Octal = 00000000015

$ BALANCE = -%X3B85
$ SHOW SYMBOL BALANCE
  BALANCE = -15237   Hex = FFFFC47B  Octal = 37777742173
```

- Fractions—A number cannot include a decimal point. In calculations, fractions are truncated; for example, 8 divided by 3 equals 2.

Numbers are stored internally as signed 4-byte integers, called longwords; positive numbers have values of 0 through 2147483647 and negative numbers have values of 4294967296 minus the absolute value of the number. The number –15237, for example, is stored as 4294952059. Negative numbers are converted back to minus-sign format for ASCII or decimal displays; however, they are not converted back for hexadecimal and octal displays. For example, the number –15237 appears in displays as hexadecimal FFFFC47B (decimal 4294952059) rather than hexadecimal –00003B85.

Numbers are stored in text files as a series of digits using ASCII conventions (for example, the digit 1 has a storage value of 49).

5.6.1.3 Values Returned by Lexical Functions

Typically used in command procedures, lexical functions provide users with a means to obtain information from the system, including information about system processes, batch and print queues, and user processes. You can also use lexical functions to manipulate character strings and translate logical names. When you assign a lexical function to a symbol, the symbol is equated to the information returned by the lexical function (for example, a number or character string). At DCL level, you can then display that information with the DCL command SHOW SYMBOL. In a command procedure, the information stored in the symbol can be used later in the procedure. See the description of the DCL commands and lexical functions in the Reference Section.

To use a lexical function, type the name of the lexical function (which always begins with F$) and its argument list. Use the following syntax:

F$function-name(args[,...])

The argument list follows the function name with any number of intervening spaces and tabs. When using a lexical function, observe the following rules:

- Enclose the argument list in parentheses.
- Within the list, specify arguments in exact order and separate them with commas; even if you omit an optional argument, do not omit the comma.
- If no arguments are required, type an empty set of parentheses.
- Do not enclose lexical functions in quotation marks.
- If an argument contains a character string, enclose the character string in quotation marks.
- If an argument contains an integer, a symbol, or another lexical function, do not enclose these values in quotation marks.

Use lexical functions the same way you would use character strings, integers, and symbols. The following example uses the F$LENGTH function. F$LENGTH returns an integer that specifies the length of the string. The returned value is assigned to the symbol LEN.

```
$ LEN = F$LENGTH("The cow jumped over the moon.")
$ SHOW SYMBOL LEN
  LEN = 29   Hex = 0000001D  Octal = 00000000035
```

You can use a lexical function in any position that you can use a symbol. In positions where symbol substitution must be forced by enclosing the symbol in apostrophes, lexical function evaluation must be forced by placing the lexical function within apostrophes. Lexical functions can also be used as argument values in other lexical functions.

The following example equates the length of the character symbol LINE to a numeric symbol named L:

```
$ L = F$LENGTH (LINE)
```

The following example strips the last two characters from the character string that is the value of the symbol LINE:

```
$ LINE = F$EXTRACT (0,F$LENGTH(LINE)-2,LINE)
```

5.6.1.4 Logical Values

Some operations interpret numbers and character strings as logical data with values as follows:

- True—A number has a logical value of true if it is odd (that is, the low-order bit is 1). A character string has a logical value of true if the first character is an uppercase or lowercase T or Y.
- False—A number has a logical value of false if it is even (that is, the low-order bit is 0). A character string has a logical value of false if the first character is not an uppercase or lowercase T or Y.

In both of the following examples, DOG_COUNT is assigned the value 13. IF STATUS means if the logical value of STATUS is true.

```
$ STATUS = 1
$ IF STATUS THEN DOG_COUNT = 13

$ STATUS = "TRUE"
$ IF STATUS THEN DOG_COUNT = 13
```

5.6.1.5 Using a Symbol as a Value for Another Symbol

After a symbol is defined, it can be used as a value for another symbol. It can be interpreted as a character string or a number, depending on the context in which it is used. For example, suppose a symbol, COUNT, is assigned the integer value 3 as follows:

```
$ COUNT = 3
```

Then the value of COUNT can be used in other assignment statements. In the following example, the value of COUNT is added to 1:

```
$ TOTAL = COUNT + 1
```

The result, 4, is equated to the symbol TOTAL. You can confirm the assignment of the value to TOTAL by entering the SHOW SYMBOL command as follows:

```
$ SHOW SYMBOL TOTAL
  TOTAL = 4    Hex = 00000004  Octal = 00000000004
```

You can include the symbol COUNT in a string assignment statement as follows:

```
$ BARK := P'COUNT'
```

COUNT is converted to a string value and appended to the character P. BARK now has the value P3.

To include a symbol in a string assignment, use either a colon and an equal sign (:=) or a colon and two equal signs (:==), and enclose the symbol in apostrophes. Otherwise, DCL will not recognize it as a symbol.

If you define a null character string for a symbol, that symbol has a value of 0, as shown in the following example:

```
$ A = ""
$ B = 2
$ C = A + B
$ SHOW SYMBOL C
   C = 2  Hex = 00000002  Octal = 00000000002
```

5.6.2 Using Symbols in Expressions

An *expression* is a combination of values. Each value in an expression is connected to another value by an *operator*. Operators are denoted in the following ways:

- Special characters—Asterisk (*), slash (/), plus sign (+), and minus sign (-).
- Special names—.EQ., .GE., .GT., .LE., .LT., .NE., .NOT., .AND., and .OR.; the names can be uppercase or lowercase.

Data entities and operators can be adjacent or can be separated by any number of spaces or tabs. The values in the expression can be symbols or literal values (such as 3 or "DOG"). Expressions take the following two forms:

- Operations—An operation combines two data entities or alters a data entity. The following example combines the literal values 10 and 3 by adding them:

  ```
  $ DOG_COUNT = 10 + 3
  $ SHOW SYMBOL DOG_COUNT
    DOG_COUNT = 13   Hex = 0000000D  Octal = 00000000015
  ```

- Comparisons—A comparison evaluates a relationship between two entities as true or false. A true comparison evaluates to a numeric value of 1, and a false comparison evaluates to a numeric value of 0. The following example compares the value of the symbol DOG_COUNT with the literal value 13 and finds them to be equal:

  ```
  $ DOG_CHECK = DOG_COUNT .EQ. 13
  $ SHOW SYMBOL DOG_CHECK
    DOG_CHECK = 1   Hex = 00000001  Octal = 00000000001
  ```

You can create character string expressions, numeric expressions, and logical expressions. These are described in the following sections.

5.6.2.1 Character String Expressions

A character string expression can contain character strings, lexical functions that evaluate to character strings, or symbols that have character string values. Attempting an operation or comparison between a character string and a number causes the character string to be converted to a number.

You can specify the following character string operations:

- Concatenation—The plus sign concatenates two character strings. For example:

```
$ COLOR = "light brown"
$ WEIGHT = "30 lbs."
$ DOG2 = "No tag, " + COLOR + ", " + WEIGHT
$ SHOW SYMBOL DOG2
  DOG2 = "No tag, light brown, 30 lbs."
```

- Reduction—The minus sign removes the second character string from the first character string. For example:

```
$ SHOW SYMBOL DOG2
  DOG2 = "No tag, light brown, 30 lbs."
$ DOG2 = DOG2 - ", 30 lbs."
$ SHOW SYMBOL DOG2
  DOG2 = "No tag, light brown"
```

  If the second character string occurs more than once in the first character string, only the first occurrence of the string is removed.

When you compare two character strings, the strings are compared character by character; strings of different lengths are not equal (for example, "dogs" is greater than "dog").

The comparison criteria are the ASCII values of the characters. Under this criterion, the digits 0 through 9 are less than the letters A through Z, and the uppercase letters A through Z are less than the lowercase letters a through z. A character string comparison terminates when either of the following conditions is true:

1. All the characters have been compared, in which case the strings are equal.
2. The first mismatch occurs.

You can specify the following varieties of string comparisons. In the examples, assume that the symbol LAST_NAME has the value "WHITFIELD."

- Equal to—The operator .EQS. compares one character string to another for equality. The following comparison evaluates to 0 to indicate that the value of the symbol LAST_NAME does not equal the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .EQS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 0   Hex = 00000000  Octal = 00000000000
```

- Greater than or equal to—The operator .GES. compares one character string to another for a greater or equal value in the first specified string. The following comparison evaluates to 1 to indicate that the value of the symbol LAST_NAME is greater than or equal to the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .GES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 1   Hex = 00000001  Octal = 00000000001
```

- Greater than—The operator .GTS. compares one character string to another for a greater value in the first specified string. The following comparison evaluates to 1 to indicate that the value of the symbol LAST_NAME is greater than the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .GTS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 1   Hex = 00000001  Octal = 00000000001
```

- Less than or equal to—The operator .LES. compares one character string to another for a lesser or equal value in the first specified string. The following comparison evaluates to 0 to indicate that the value of the symbol LAST_NAME is not less than or equal to the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .LES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 0   Hex = 00000000  Octal = 00000000000
```

- Less than—The operator .LTS. compares one character string to another for a lesser value in the first specified string. The following comparison evaluates to 0 to indicate that the value of the symbol LAST_NAME is not less than the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .LTS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 0   Hex = 00000000  Octal = 00000000000
```

- Not equal—The operator .NES. compares one character string to another for inequality. The following comparison evaluates to 1 to indicate that the value of the symbol LAST_NAME does not equal the literal "NORMAN":

```
$ TEST_NAME = LAST_NAME .NES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
  TEST_NAME = 1   Hex = 00000001  Octal = 00000000001
```

5.6.2.2 Numeric Expressions

In a numeric expression, the values involved must be literal numbers (such as 3) or symbols with numeric values. In addition, you can use a character string that represents a number (for example, "23" or "-51"). Attempting an operation or comparison between a number and a character string causes the character string to be converted to a number.

You can specify the following numeric operations:

- Multiplication—The asterisk multiplies two numbers. For example:

```
$ BALANCE = 142 * 14
$ SHOW SYMBOL BALANCE
  BALANCE = 1988    Hex = 000007C4  Octal = 00000003704
```

- Division—The slash divides the first specified number by the second specified number. For example:

```
$ BALANCE = BALANCE / 14
$ SHOW SYMBOL BALANCE
  BALANCE = 142    Hex = 0000008E  Octal = 00000000216
```

  If a number does not divide evenly, the remainder is lost. (No rounding takes place.)

- Addition—The plus sign adds two numbers. For example:

```
$ BALANCE = BALANCE + 37
$ SHOW SYMBOL BALANCE
  BALANCE = 179    Hex = 000000B3  Octal = 00000000263
```

- Subtraction—The minus sign subtracts the second specified number from the first specified number. For example:

```
$ BALANCE = BALANCE - 15416
$ SHOW SYMBOL BALANCE
  BALANCE = -15237    Hex = FFFFC47B  Octal = 00000142173
```

- Unary plus and minus—The plus and minus signs change the sign of the number they precede. For example:

```
$ BALANCE = -(-142)
$ SHOW SYMBOL BALANCE
  BALANCE = 142    Hex = 0000008E    Octal = 00000000216
```

You can specify the following numeric comparisons:

- Equal to—The operator .EQ. compares one number to another for equality. The following comparison evaluates to 1 to indicate that BALANCE equals –15237:

```
$ TEST_BALANCE = BALANCE .EQ. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1    Hex = 00000001  Octal = 00000000001
```

- Greater than or equal to—The operator .GE. compares one number to another for a greater or equal value in the first number. The following comparison evaluates to 1 to indicate that BALANCE is greater than or equal to -15237:

```
$ TEST_BALANCE = BALANCE .GE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1   Hex = 00000001  Octal = 00000000001
```

- Greater than—The operator .GT. compares one number to another for a greater value in the first number. The following comparison evaluates to 0 to indicate that BALANCE is not greater than -15237:

```
$ TEST_BALANCE = BALANCE .GT. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000  Octal = 00000000000
```

- Less than or equal to—The operator .LE. compares one number to another for a lesser or equal value in the first number. The following comparison evaluates to 1 to indicate that BALANCE is less than or equal to -15237:

```
$ TEST_BALANCE = BALANCE .LE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1   Hex = 00000001  Octal = 00000000001
```

- Less than—The operator .LT. compares one number to another for a lesser value in the first number. The following comparison evaluates to 0 to indicate that BALANCE is not less than -15237:

```
$ TEST_BALANCE = BALANCE .LT. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000  Octal = 00000000000
```

- Not equal to—The operator .NE. compares one number to another for inequality. The following comparison evaluates to 0 to indicate that BALANCE equals -15237:

```
$ TEST_BALANCE = BALANCE .NE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000  Octal = 00000000000
```

5.6.2.3 Logical Expressions

A logical operation affects all the bits in the number being acted upon. The values for logical expressions are integers, and the result of the expression is an integer as well. If you specify a character string value in a logical expression, the string is converted to an integer before the expression is evaluated.

String and integer values are evaluated as follows:

- If the first character is T, t, Y, or y, a character string has a logical value of true (1).
- If the first character is not T, t, Y, or y, a character string has a logical value of false (0).
- If an integer is odd (the low-order bit is 1), it has a logical value of true (1).

- If an integer is even (the low-order bit is 0), it has a logical value of false (0).

Typically, you use logical expressions to evaluate the low-order bit of a logical value; that is, to determine whether the value is true or false. You can specify the following logical operations:

- Not—The operator .NOT. reverses the bit configuration of a logical value. A true value becomes false and a false value becomes true. The following example reverses a true value to false. The expression evaluates to –2; the value is even and is therefore false:

```
$ SHOW SYMBOL STATUS
  STATUS = 1    Hex = 00000001  Octal = 00000000001
$ STATUS = .NOT. STATUS
$ SHOW SYMBOL STATUS
  STATUS = -2   Hex = FFFFFFFE  Octal = 37777777776
```

- And—The operator .AND. combines two logical values as follows:

Bit Level	Entity Level
1 .AND. 1 = 1	true .AND. true = true
1 .AND. 0 = 0	true .AND. false = false
0 .AND. 1 = 0	false .AND. true = false
0 .AND. 0 = 0	false .AND. false = false

The following example combines a true value and a false value to produce a false value:

```
$ STAT1 = "TRUE"
$ STAT2 = "FALSE"
$ STATUS = STAT1 .AND. STAT2
$ SHOW SYMBOL STATUS
  STATUS = 0    Hex = 00000000  Octal = 00000000000
```

- Or—The operator .OR. combines two logical values as follows:

Bit Level	Entity Level
1 .OR. 1 = 1	true .OR. true = true
1 .OR. 0 = 1	true .OR. false = true
0 .OR. 1 = 1	false .OR. true = true
0 .OR. 0 = 0	false .OR. false = false

The following example combines a true value and a false value to produce a true value:

```
$ STAT1 = "TRUE"
$ STAT2 = "FALSE"
$ STATUS = STAT1 .OR. STAT2
$ SHOW SYMBOL STATUS
  STATUS = 1   Hex = 00000001  Octal = 00000000001
```

5.6.2.4 Substring Replacement and Numeric Overlays

You can replace a part of a character string with another character string. The assignment statement has the following format:

symbol-name[offset,size] := replacement-string

or

symbol-name[offset,size] := replacement-string

The fields are as follows:

- *Offset* is an integer that indicates the position of the replacement-string relative to the first character in the original string. An offset of 0 means the first character in the symbol, an offset of 1 means the second character, and so on.
- *Size* is an integer that indicates the length of the replacement-string.

To replace substrings, observe the following rules:

- The square brackets are required notation. No spaces are allowed between the symbol name and the left bracket.
- Integer values can be in the range of 0 through 768.
- The replacement-string must be a character string.

In the following example, the first assignment statement gives the symbol A the value PACKRAT. The second statement specifies that MUSK replace the first four characters in the value of A. The result is that the value of A becomes MUSKRAT.

```
$ A := PACKRAT
$ A[0,4] := MUSK
$ SHOW SYMBOL A
   A = "MUSKRAT"
```

The symbol name you specify can be undefined initially. The assignment statement creates the symbol name and, if necessary, provides leading or trailing spaces in the symbol value. For example:

```
$ B[4,3] := RAT
```

If the symbol B does not have a previous value, it is given a value of four leading spaces followed by RAT. This format creates a blank line of any length. The following example gives the symbol LINE a value of 80 blank spaces:

```
$ LINE[0,80]:= " "
```

Lining up records in columns makes a list easier to read and sort. You can use this format to specify how you want data to be stored. For example:

```
$ DATA[0,15] := 'NAME'
$ DATA[17,1] := 'GRADE'
```

The first statement fills in the first 15 columns of DATA with whatever value NAME has. The second statement fills in column 18 with whatever value GRADE has. Columns 16 and 17 contain blanks.

A special format of the assignment statement can also be used to perform binary (bit-level) overlays of the current symbol value. This format is as follows:

$ symbol-name[bit-position,size] = replacement-expression

or

$ symbol-name[bit-position,size] = replacement-expression

where:

- *bit-position* is an integer that indicates the location relative to bit 0 at which the overlay is to occur.
- *size* is an integer that indicates the number of bits to be overlaid.

When using numeric overlays, observe the following rules:

- The square brackets are required notation. No spaces are allowed between the symbol name and the left bracket.
- Literal values are assumed to be decimal.
- The maximum length for both *bit-position* and *size* is 32 bits.
- The *replacement-expression* must be a numeric expression.
- When *symbol-name* is either undefined or defined as a string, the result of the overlay is a string. Otherwise, the result is an integer.

The following example defines the symbol BELL as the value 7. The low-order byte of BELL has the binary value 00000111. By changing the 0 at offset 5 to 1 (beginning with 0, count bits from right to left), you create the binary value 00100111 (decimal value 39).

```
$ BELL = 7
$ BELL[5,1] = 1
$ SHOW SYMBOL BELL
 BELL = 39   Hex = 00000027   Octal = 00000000047
```

5.6.2.5 Order of Operations and the Results of Evaluations

An expression can contain any number of operations and comparisons. You can indicate precedence (the order in which operation and comparison should be evaluated) by placing operations to be performed first in parentheses. (Parentheses can be nested.) Otherwise, operations within an expression are evaluated in the following order:

1. Unary plus (+) and minus (-)
2. Multiplication and division
3. All other numeric and character operations
4. All numeric and character comparisons
5. Logical NOT operations
6. Logical AND operations
7. Logical OR operations

Operations and comparisons that have the same precedence are evaluated from left to right. The following examples illustrate precedence of operations in expressions:

```
$ BALANCE = 150 + 20 * 4
      (BALANCE = 150 + 80)
$ SHOW SYMBOL BALANCE
  BALANCE = 230    Hex = 000000E6  Octal = 00000000346

$ BALANCE = (150 + 20) * 4
  (BALANCE = 170 * 4)
$ SHOW SYMBOL BALANCE
  BALANCE = 680  Hex = 000002A8  Octal = 00000001250

$ STATUS = 150 * 4 .GT. 80 * 2
      (STATUS = 600 .GT. 160)
$ SHOW SYMBOL STATUS
  STATUS = 1    Hex = 00000001  Octal = 00000000001
```

An expression has either an integer or a string value, depending on the types of values and the operators used. Table 5–2 summarizes how DCL evaluates expressions. The first column lists the different values and operators that an expression might contain. The second column tells, for each case, what the entire expression is equated to. Within the table *any value* stands for a string or an integer.

Table 5-2: Determining the Value of an Expression

Expression	Resulting Value Type
Integer value	Integer
String value	String
Integer lexical function	Integer
String lexical function	String
Integer symbol	Integer
String symbol	String
+, –, or .NOT. any value	Integer
Any value .AND. or .OR. any value	Integer
String + or—string	String
Integer + or—any value	Integer
Any value + or—integer	Integer
Any value * or / any value	Integer
Any value (string comparison) any value	Integer
Any value (numeric comparison) any value	Integer

Chapter 6

Writing and Using Command Procedures

A command procedure is a file that contains DCL commands and data lines used by the DCL commands. You can write both simple and complex command procedures. A simple command procedure executes a series of DCL commands in the order in which they are written. A complex command procedure performs program-like functions.

6.1 Format

Follow these instructions when formatting a command procedure:

- Begin each line containing a command, comment, or label with a dollar sign.
- Do not begin data lines with a dollar sign.
- Use comments to explain the command procedure to anyone who must maintain it. Place comments at the beginning of a procedure to describe the procedure and the parameters passed to it; place them at the beginning of each block of commands to describe that section of the procedure. The command interpreter ignores comments when the command procedure executes. Precede a comment with an exclamation point; the comment is all text to the right of an exclamation point. (To include a literal exclamation point in a command line, precede and follow it with quotation marks.)
- Use complete names for commands and qualifiers. Commands and qualifiers are usually self-explanatory when they are not abbreviated. Abbreviated commands and qualifiers may no longer be unique when new commands and qualifiers are added to the VMS operating system.
- Put labels on separate lines to make loops, subroutines, and conditional code easier to understand. (Labels mark the beginning of loops, subroutines, and conditional code.) You may choose to differentiate labels from commands by placing labels immediately after the dollar sign and preceding commands by a space. A label can have up to 255 characters, cannot contain embedded blanks, and must be terminated by a colon. (The GOTO, GOSUB, and CALL commands transfer control to labels, which mark the beginning of a loop, a section of code, or a subroutine.)

- Separate command sequences with lines containing a dollar sign and an exclamation point ($!). This makes it easier to see the outline of the command procedure. (If you insert blank lines, the command interpreter interprets them as data lines and produces a message warning you that the data lines were ignored. If you insert lines containing only a dollar sign, the command interpreter searches the whole line for a command.)

6.2 Execution

Command procedures can be executed interactively from DCL level, from within another command procedure, on a remote node using the TYPE command, or in batch mode. To execute a command procedure interactively, type an at sign (@) followed by the file specification of the procedure. The file type defaults to COM. The following command executes the procedure SETD.COM in the directory [MAINT.PROCEDURES] on the disk WORKDISK:

```
$ @WORKDISK:[MAINT.PROCEDURES]SETD
```

To simplify the invocation of a procedure, create a global symbol or a logical name, and place the symbol or logical name in your login command procedure. (Section 6.3 describes how to create a login command procedure. Symbols are described in Chapter 5. See Chapter 4 for more information about logical names.) Equating the command line to a global symbol allows you to invoke the command procedure from any directory by entering the global symbol name as follows:

```
$ SETD == "@WORKDISK:[MAINT.PROCEDURES]SETD"
$ SETD
```

Equating the file specification to a logical name allows you to invoke the command procedure from any directory by entering an at sign followed by the logical name as follows:

```
$ DEFINE SETD WORKDISK:[MAINT.PROCEDURES]SETD.COM
$ @SETD
```

To invoke a command procedure from within another command procedure, use the at sign (@) followed by the file specification of the procedure. In the following example, the command procedure WRITEDATE.COM invokes the command procedure GETDATE.COM:

```
$! WRITEDATE.COM
$ INQUIRE TIME "What is the current time, in hh:mm format?"
$ @GETDATE
```

Use the TYPE command to execute a command procedure interactively on a remote node. The TYPE command lets you execute command procedures to list the users logged on to the remote node or to display the status of services in the local cluster not provided clusterwide. The output of the command procedure is displayed on the user's terminal at the local node.

To execute a command procedure in the default DECnet account of the remote node, specify the command procedure as a parameter to the TYPE command as follows:

```
$ TYPE node_name::"TASK=command_procedure"
```

The variable *node_name* is the name of the remote node on which the command procedure resides; *command_procedure* is the name of the command procedure.

To execute a command procedure in the top level directory of another account on the remote node, use an access control string in the command as follows:

```
$ TYPE node_name"user_name password"::"TASK=command_procedure"
```

The variable *user_name* is the user name of the account on the remote node, *password* is the password of the account on the remote node, and *command_procedure* is the name of the command procedure.

The following command procedure, SHOWUSERS.COM, displays the users logged in at the remote node on which the command procedure resides:

```
$! SHOWUSERS.COM
$ IF F$MODE() .EQS. "NETWORK" THEN DEFINE/USER SYS$OUTPUT SYS$NET
$ SHOW USERS
```

The following command executes the command procedure SHOWUSERS.COM and displays the output from this command procedure on the user's terminal. The command procedure resides in the top level directory of account BIRD on node ORIOLE.

```
$ TYPE ORIOLE"BIRD FLIESFAST"::"TASK=SHOWUSERS"

          VAX/VMS Interactive Users
           09-DEC-1988 17:20:13.30
    Total number of interactive users = 4
 Username      Process Name       PID      Terminal
 FLICKER       Freddie         00536278    TXA1:
 ROBIN         Red             00892674    VTA2:
 DOVE          Whitie          00847326    TXA3:
 DUCK          Donna           02643859    RTA1:
```

You can also submit a command procedure to a batch queue to execute as a batch job. If your system is part of a network, you can submit a command procedure to execute as a batch job on a remote node. Within a command procedure, you can use DCL commands to open and close files on a remote node and read and write records in these files, using the same commands and qualifiers as for local files. Section 3.1.4 contains more information about batch jobs.

6.2.1 Changing Command Levels

A command level is an input stream for the DCL command interpreter. You can create a maximum of 32 command levels. There are two ways to create new command levels. You can either use the CALL command to call a subroutine that exists within the command procedure, or you can nest command procedures by using an execute procedure (@) command inside one command procedure to invoke another command procedure. When you use the CALL command or nest a command procedure, the command level increases by 1.

When you invoke a command procedure, the command level increases by 1. For example, if you invoke procedure SUB from DCL command level (level 0), SUB executes at command level 1. If SUB then invokes SUB1, which invokes SUBSUB1, SUB1 executes at command level 2, and SUBSUB1 executes at command level 3.

By convention, DCL level (command level 0) is the highest command level and command level 31 the lowest command level. Thus, when you move from command level 3 to command level 2, you are moving to the next higher command level.

6.2.2 Exiting from Command Procedures

A command procedure exits when it reaches the end of the procedure, an EXIT command, or a STOP command. If the exit is caused by the end of the procedure or an EXIT command, control returns to the next higher command level. If the exit is caused by the EXIT command, you can return a status value to the next higher command level by specifying the value as the parameter of the EXIT command. (A status value is a hexadecimal representation of a VMS message code.) This status value is placed in the global symbol $STATUS. If you return the status value 44 with the EXIT command, control returns to DCL command level and the following error message is displayed:

```
%SYSTEM-F-ABORT, abort
```

See Section 5.2 for more information about the global symbols $STATUS and $SEVERITY. For example, if you invoke SUB at DCL command level, and SUB calls SUB1, the following sequence of actions occurs:

1. Exiting from SUB1 returns you to SUB at the command line following the call to SUB1.
2. Exiting from SUB returns you to DCL command level.

If the exit is caused by the STOP command, control always returns to DCL command level, regardless of the command level in which the STOP command executes.

6.3 Designing a Login Command Procedure

You can create a command procedure, called a login command procedure, to execute the same commands each time you log in. Name your login command procedure LOGIN.COM, and place it in your top level directory, unless your system manager tells you otherwise.

The following sample LOGIN.COM procedure illustrates some commands you may want to include in your login command procedure:

```
$! Sample LOGIN.COM for user MARCIA with
$!  default disk of DISK3
$!
$! Exit if this is a batch job or another
$! type of noninteractive process
$!
$ IF F$MODE() .NES. "INTERACTIVE" THEN EXIT ❶
$!
$! Tailor the default behavior of
$! certain DCL commands
$!
$ PUR*GE :== PURGE/LOG
$ SUB*MIT :== SUBMIT/NOLOG_FILE/NOTIFY
$ M*AIL :== MAIL/EDIT=(SEND,FORWARD,REPLY)
$!
$! Define global symbols
$!
$ DISPLAY :== MONITOR PROCESSES/TOPCPU
$ GO :== SET DEFAULT
$ LP :== SHOW QUEUE/ALL SYS$PRINT
$ SS :== SHOW SYMBOL
$ SQ :== SHOW QUEUE/ALL
$ REM :== @DISK3:[MARCIA.PROG]REMINDER
$ MAIN :== SET DEFAULT DISK3:[MARCIA]
$!
$! Define logical names for:
$!   Directories
$ DEFINE HOME DISK3:[MARCIA]
$ DEFINE REV DISK3:[MARCIA.REVIEWS]
$ DEFINE TOOLS DISK3:[MARCIA.TOOLS]
$!   Files
$ DEFINE EQUIP DISK3:[MARCIA.LISTS]EQUIPMENT.DAT
$ DEFINE ACCOMP DISK3:[MARCIA]ACCOMPLISHMENTS.DAT
$!   Users
$ DEFINE JON DAISY::HARRIS
$ DEFINE JANE DAISY::MOORE
$!
$! Define keys to execute commands
$!
$ DEFINE/KEY PF3 "SHOW USERS" /TERMINATE
$ DEFINE/KEY KP7 "SPAWN" /TERMINATE
$ DEFINE/KEY KP8 "ATTACH "
$ DEFINE/KEY KP4 "SET HOST "
$!
$! Change the prompt string to a three-character
$! abbreviation of the node name
$!
```

```
$ NODE = F$GETSYI("NODENAME")  ❷
$ PROMPT = F$EXTRACT(0,3,NODE)
$ SET PROMPT = "''PROMPT'> "
$!
$! Type the system notices  ❸
$!
$ TYPE SYS$SYSTEM:NOTICE.TXT
$!
$! Run a program that displays today's appointments  ❹
$!
$ RUN DISK3:[MARCIA.PROG]REMINDER
```

❶ The F$MODE lexical function returns the mode (interactive, batch, network, or other) that the process is in when the LOGIN.COM procedure is executing. This statement causes the procedure to exit unless you are using the system interactively. You should test the mode at the beginning of your LOGIN.COM procedure to ensure that commands used only in interactive mode are not executed in any other mode; in some cases, these commands can abort noninteractive processes.

❷ This group of commands changes the DCL prompt to the first three characters of the node name. The F$GETSYI lexical function determines the node name. The F$EXTRACT lexical function extracts the first three characters of the name. The SET PROMPT command changes the prompt from a dollar sign to the first three characters of the node name followed by the right angle bracket character (>) and a space.

❸ This command displays the system notices that your system manager keeps in the file SYS$SYSTEM:NOTICE.TXT.

❹ This command runs a user-written program that displays your daily appointments. If you have written programs that you always run after you log in, you may prefer to execute them directly from your LOGIN.COM file.

The system manager assigns the file specification for your login command procedure in the LGICMD field for your account. In most installations, the login command procedure is called LOGIN.COM. However, if you want to execute a file other than the one named in the LGICMD field for your account, use the /COMMAND qualifier when you log in.

6.4 Passing Data

Command procedures frequently require data provided by a user. To specify the same data each time the command procedure is executed, place the data on data lines following the command that requires the data. (A data line is a line that does not begin with a dollar sign. To include a data line that begins with a dollar sign, use the DCL commands DECK and EOD, which are described in the Reference Section.) The following command procedure executes the image CENSUS.EXE, which reads the data 1981, 1982, and 1983 each time the procedure is executed:

```
$ ! CENSUS.COM
$ !
$ RUN CENSUS
1981
1982
1983
$ EXIT
```

The text on a data line is passed directly to the image; DCL does not process data lines. Therefore, DCL does not translate symbols or evaluate arithmetic expressions on data lines before passing the symbols or arithmetic expressions to the image. Logical names are not translated by DCL; therefore, a logical name included on a data line is translated before it is passed to an image.

To specify different data each time a command procedure executes, use one of the following mechanisms, which are described in Sections 6.4.1 through 6.4.4:

- Pass the data as one or more parameter values.
- Use the INQUIRE or READ command within the command procedure to prompt for data.
- Specify a device or file from which to read the data by redefining the logical name SYS$INPUT.

6.4.1 Using Parameters to Pass Data

When you invoke a command procedure, you can pass it up to eight parameters. Place the parameters after the file specification of the command procedure. Separate the parameters with one or more spaces or tabs. For example, the following command invokes SUM.COM and passes eight parameters to the procedure:

```
$ @SUM 34 52 664 89 2 7 87 3
```

To pass parameters to a command procedure executed in batch mode, use the /PARAMETERS qualifier of the SUBMIT command. If you pass more than one parameter, place the parameters in parentheses and separate them with commas. If you execute more than one command procedure using a single SUBMIT command, the specified parameters are used for each command procedure in the batch job. The following command passes three parameters to the command procedures ASK.COM and GO.COM, which are executed as batch jobs:

```
$ SUBMIT/PARAMETERS=(TODAY,TOMORROW,YESTERDAY) ASK.COM, GO.COM
```

DCL places parameters passed to a command procedure in the local symbols P1 through P8; P1 is assigned the first parameter value; P2 the second; P3 the third and so on. If you pass more than eight values, you receive the following error message and the command procedure does not execute:

```
%DCL-W-DEFOVF, too many command procedure parameters - limit to eight
```

If you pass fewer than eight values, the extra symbols are assigned null values.

Specify a parameter value as one of the following:

- Integer—When you specify an integer, it is converted to a string as follows:

  ```
  $ @ADDER 24 25
  ```

  In this example, P1 is the string value 24; P2 is the string value 25. (You can, however, use the symbols P1 and P2 in both integer and character string expressions; DCL performs the necessary conversions automatically.)

- String—Specify character strings as follows:

  ```
  $ @DATA Paul Cramer
  ```

  In this example, the strings Paul and Cramer are converted to uppercase letters; P1 is PAUL and P2 is CRAMER.

  To preserve spaces, tabs, or lowercase characters, place quotation marks before and after the string as follows:

  ```
  $ @DATA "Paul Cramer"
  ```

  In this example, P1 is Paul Cramer and P2 is null.

- Symbol—To pass the value of a symbol, place an apostrophe character before and after the symbol, as shown in the following example. When passing a symbol, DCL removes quotation marks that enclose a string. (To preserve spaces, tabs, and lowercase characters in a symbol value, surround the symbol with quotation marks.)

  ```
  $ NAME = "Paul Cramer"
  $ @DATA 'NAME'
  ```

  In this example, P1 is Paul and P2 is Cramer.

  To include a quotation mark as part of a string, enter three quotation marks as follows:

  ```
  $ NEW_NAME = """Paul Cramer"""
  $ @DATA 'NEW_NAME'
  ```

  In this example, P1 is "Paul Cramer" and P2 is null.

- Null—To pass a null parameter, use a set of quotation marks as a placeholder in the command string. In the following example, the first parameter passed to DATA.COM is a null parameter:

  ```
  $ @DATA "" "Paul Cramer"
  ```

In the preceding example, P1 is null, and P2 is Paul Cramer.

For example, when DATA.COM is invoked with the following command, P1 through P8 are defined in DATA.COM as follows:

P1 = Paul Cramer
P2 = 24
P3 = (555) 111-1111
P4–P8 = null

```
$ @DATA  "Paul Cramer" 24 "(555) 111-1111"
```

You can pass up to eight parameters to a nested command procedure. The local symbols P1 through P8 in the nested procedure are not related to the local symbols P1 through P8 in the invoking procedure. In the following example, DATA.COM invokes the nested command procedure NAME.COM:

```
$ ! DATA.COM
$ @NAME 'P1' Joe Cooper
```

Because P1 in DATA.COM is the string Paul Cramer, which contains no quotation marks, it is passed to NAME.COM as two parameters. In NAME.COM, P1 through P8 are defined as follows:

P1 = PAUL
P2 = CRAMER
P3 = JOE
P4 = COOPER
P5–P8 = null

Because DCL removes quotation marks when passing a symbol, you must enclose the value in three sets of quotation marks to preserve spaces, tabs, and lowercase characters in the symbol value. In the following example, the literal value in P1 is enclosed in three sets of quotation marks and passed to NAME.COM. If P1 originally contained the value "Paul Cramer", the value "Paul Cramer" is passed to NAME.COM.

```
$ ! DATA.COM
$ QUOTE = """
$ P1 = QUOTE + P1 + QUOTE
$ @NAME 'P1' "Joe Cooper"
```

In this example, P1 is Paul Cramer and P2 is Joe Cooper in the command procedure NAME.COM.

An alternative is to enclose the text in quotation marks and, where a symbol appears, precede the symbol with two apostrophes and follow it with one apostrophe as follows:

```
$ ! DATA.COM
$ @NAME "''P1'"
```

Passing Data and Parameters to a Batch Job

To specify parameters for a job submitted in batch mode, use the /PARAMETERS qualifier of the SUBMIT command. Note that you can also pass data to a batch job by including the data in a command procedure or by defining SYS$INPUT to be a file. The specified parameters are used for each command procedure in the batch job. The following SUBMIT command passes two parameters to the command procedures LIBRARY.COM and SORT.COM:

```
$ SUBMIT-
_$ /PARAMETERS=(DISK:[ACCOUNT.BILLS]DATA.DAT,DISK:[ACCOUNT]NAME.DAT) -
_$ LIBRARY.COM, SORT.COM
```

Your batch job executes as if you had logged in and executed each of the submitted command procedures. For example, the previous SUBMIT command executes a batch job that logs in under your account, executes your login command procedure, and then executes the following commands:

```
$ @LIBRARY DISK:[ACCOUNT.BILLS]DATA.DAT DISK:[ACCOUNT]NAME.DAT
$ @SORT DISK:[ACCOUNT.BILLS]DATA.DAT DISK:[ACCOUNT]NAME.DAT
```

6.4.2 The INQUIRE Command

You can use the INQUIRE command to obtain data for command procedures that you execute interactively. The INQUIRE command prompts for a value, reads the value from the terminal, and assigns it to a symbol. The response to the prompt is interpreted as a character string. By default, the response is converted to uppercase, multiple blanks and tabs are replaced by a single space, and leading and trailing spaces are removed. To preserve lowercase characters, multiple spaces, and tabs, enclose your response in quotation marks. The following command procedure writes the prompt *Filename:* and puts your response into the local symbol FILE:

```
$ INQUIRE FILE "Filename"
```

To suppress the colon and space automatically added to the end of the prompt, use the /NOPUNCTUATION qualifier. To make the symbol global instead of local, use the /GLOBAL qualifier. The following command procedure writes the prompt *Do you want to use defaults?* and puts the response into the global symbol DEFAULT:

```
$ INQUIRE/NOPUNCTUATION/GLOBAL DEFAULT-
_$ "Do you want to use defaults?"
```

When a command procedure is submitted as a batch job, the value for a symbol specified in an INQUIRE command is read from the data line following the INQUIRE command. If you do not include a data line, the symbol is assigned a null value.

6.4.3 The READ Command

You can use the READ command to obtain data for command procedures that you execute interactively. The READ command prompts for a value, reads the value from the source specified by the first parameter, and assigns it to the symbol named as the second parameter. If you do not specify a prompt, the READ command outputs *DATA:* as the default prompt. To specify a different prompt, use the /PROMPT qualifier. All characters typed on the terminal in response to the prompt are taken as an exact character string value (case, spaces, and tabs are preserved). The following command writes the prompt *Filename:*, reads the response from the source specified by the logical name SYS$COMMAND (by default, the terminal), and assigns the response to the symbol FILE:

```
$ READ/PROMPT="Filename: "  SYS$COMMAND FILE
```

6.4.4 Obtaining Data from SYS$INPUT

Commands, utilities, and other system images usually take their input from the source specified by the logical name SYS$INPUT. SYS$INPUT is a process-permanent logical name that the system defines automatically. You can specify SYS$INPUT as any one of the following:

- Data line—In a command procedure, the default value of SYS$INPUT is the data lines of the procedure. In the following command procedure, the image CENSUS.EXE uses the default value of SYS$INPUT to take input (1986, 1987, and 1988) from the data lines:

  ```
  $ ! CENSUS.COM
  $ !
  $ ! Execute CENSUS
  $ RUN CENSUS
  1986
  1987
  1988
  $
  ```

- Terminal—A command procedure can get input from a terminal by defining SYS$INPUT as the terminal. This allows you to perform interactive tasks from a command procedure. The following command procedure defines SYS$INPUT as SYS$COMMAND, which is, by default, the terminal. The command procedure then invokes the EDT editor, beginning an interactive editing session. (The /USER_MODE qualifier redefines SYS$INPUT for a single image; you should use this qualifier whenever you redefine a process-permanent logical name.)

  ```
  $ ! EDIT.COM
  $ !
  $ ! Edit the file STATS.DAT
  $ WRITE SYS$OUTPUT "Edit STATS.DAT:"
  $ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND:
  $ EDIT STATS.DAT
  ```

- File—A command procedure can get input from a file by defining SYS$INPUT as a file. The following command procedure defines SYS$INPUT as the file YEARS.DAT, then invokes the program CENSUS. CENSUS reads its input from the file YEARS.DAT.

```
$ ! CENSUS.COM
$ !
$ ! Execute CENSUS
$ DEFINE/USER_MODE SYS$INPUT YEARS.DAT
$ RUN CENSUS
```

6.5 Returning Data

To return a value from a command procedure (either to a calling procedure or to DCL command level), you must assign the value to a global symbol. The global symbol can be read at any command level. Use comments to explain the use of any global symbols.

To create a global symbol, specify the value to be passed on the right side of a global assignment statement. In the following example, the command procedure DATA.COM invokes the command procedure NAME.COM, passing NAME.COM a full name. NAME.COM places the last name in the global symbol LAST_NAME. When NAME.COM completes, DCL continues executing DATA.COM, which reads the last name by specifying the global symbol LAST_NAME. (The command procedure NAME.COM would be in a separate file; it is indented here for clarity.)

```
$ @DATA "Paul Cramer"

$ ! DATA.COM
$ !
$ ! P1 is a full name
$ ! NAME.COM returns the last name in the
$ ! global symbol LAST_NAME
$ !
$ @NAME 'P1'
        $ ! NAME.COM
        $ ! P1 is a first name
        $ ! P2 is a last name
        $ ! return P2 in the global symbol LAST_NAME
        $ LAST_NAME == P2
        $ EXIT
$ ! write LAST_NAME to the terminal
$ WRITE SYS$OUTPUT "LAST_NAME = ''LAST_NAME'"

LAST_NAME = CRAMER
```

6.6 Displaying Data

Commands, utilities, and other system images normally write their output to the source specified by the logical name SYS$OUTPUT. By default, SYS$OUTPUT is equated to the terminal. However, you can redirect the output of a command procedure to a file by using the /OUTPUT qualifier. In the following example, output from the command procedure SETD.COM is written to the file RESULTS.TXT instead of to the terminal:

```
$ @SETD/OUTPUT=RESULTS.TXT
```

DCL commands that accept the /OUTPUT qualifier include: ACCOUNTING, CALL, DIRECTORY, HELP, LIBRARY, RUN (process), SPAWN, and TYPE.

6.6.1 Displaying Character Strings and Symbols

To display character strings and symbols on the terminal, use the WRITE command as follows:

- Character string—Enclose the text to be displayed in quotation marks. The following example displays the text: *Two files were written.*

  ```
  $ WRITE SYS$OUTPUT "Two files were written."
  ```

- Symbol value—The WRITE command automatically substitutes symbols and lexical functions. The following example displays the text *STAT1.DAT*, which is the translation of the symbol FILE:

  ```
  $ FILE =  "STAT1.DAT"
  $ WRITE SYS$OUTPUT FILE
  ```

- Combination of character strings and symbol values—Enclose the text to be displayed in quotation marks. Preface a symbol with two apostrophes, and follow it with one apostrophe. The following example displays the text: *STAT1.DAT and STAT2.DAT were written.* STAT1.DAT is the translation of the symbol AFILE; STAT2.DAT is the translation of the symbol BFILE.

  ```
  $ AFILE = "STAT1.DAT"
  $ BFILE = "STAT2.DAT"
  $ WRITE SYS$OUTPUT "''AFILE' and ''BFILE' were written."
  ```

  You can also use commas and quotation marks to display a combination of character strings and symbol values. The following example displays the same text as the previous example:

  ```
  $ AFILE = "STAT1.DAT"
  $ BFILE = "STAT2.DAT"
  $ WRITE SYS$OUTPUT AFILE, " and " ,BFILE, " were written."
  ```

6.6.2 Displaying Text

To display text that is more than one line long, use the TYPE command. TYPE writes data to SYS$OUTPUT (the terminal, by default). Using SYS$INPUT as the parameter causes TYPE to read the data from the command procedure. When the following command procedure is executed, the text on the data lines is displayed on the terminal:

```
$ ! CLEAN.COM
$ !
$ TYPE SYS$INPUT

This command procedure executes a command that allows you
to clean up a directory.

Please enter one of the following commands after the prompt:
  EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY

$ INQUIRE COMMAND "Command"
    .
    .
    .
```

6.6.3 Displaying Files

To display the contents of a file, use the TYPE command. The following example displays the file STAT1.DAT on the terminal:

```
$ TYPE DUA0:[HORACE]STAT1.DAT
```

6.7 Reading and Writing Files (File I/O)

To move data to and from files, use the OPEN, CLOSE, READ, and WRITE commands. The logical name you specify in the OPEN command is used to refer to the file in the WRITE, READ, and CLOSE commands.

6.7.1 Specifying Files in Batch Job Command Procedures

A batch job command procedure executes as if you had logged in and executed the command procedure interactively. Since your login default directory is not usually the default directory needed to access files mentioned in a command procedure, command procedures that will be executed in batch mode should use one of the following mechanisms to ensure that the correct files are accessed:

- Use complete file specifications—When specifying a file in a command procedure or passing a file to a command procedure, include the device and directory names as part of the file specification, as shown in the previous example.
- Use the SET DEFAULT command—Before accessing a file in a command procedure, use the SET DEFAULT command to specify the proper device and directory.

6.7.2 Writing to a File

To write data to a file, take the following steps:

1. Open the file—The OPEN command assigns to the logical name specified in the first parameter the file name specified in the second parameter.

   Use the /APPEND qualifier of the OPEN command to write data to the end of an existing file. If you use the /APPEND qualifier to open a nonexistent file, an error occurs and no file is opened.

   Use the /WRITE qualifier of the OPEN command to create a new file and to open this file for write access. If you use the /WRITE qualifier to open an existing file, a new version of that file is created.

2. Begin the write loop with a label—File I/O is always done in a loop unless you are writing or reading a single record.

3. Read the data to be written—Use the INQUIRE command or the READ command to read data into a symbol.

4. Test the data—Check the symbol containing the data. If the symbol is null (you pressed RETURN and entered no data on the line), you have reached the end of the data to be written to the file and should go to the end of the loop. Otherwise, continue.

5. Write the data to the file—Use the WRITE command to write the value of the symbol (one record) to the file.

6. Return to the beginning of the loop—You remain in the loop until there is no more data to be written to the file.

7. End the loop and close the file—The CLOSE command disassociates the file name from the logical name and closes the file. (Files opened by the OPEN command remain open until you log out unless you explicitly close them.)

The following command procedure writes data to the new file STAT.DAT. If a file of that name exists, a new version is created.

```
$ ! Write a file
$ ON ERROR THEN EXIT                        !EXIT if the command procedure
$                                           ! cannot open the file
$ OPEN/WRITE IN_FILE STAT.DAT               !Open the file
$ ON CONTROL_Y THEN GOTO END_WRITE          !Close the file if you abort
$                                           ! execution with a CTRL/Y
$ ON ERROR THEN GOTO END_WRITE              !Close the file if an error
$                                           ! occurs
$WRITE:                                     !Begin loop
$ INQUIRE STUFF "Input data"                !Get input
$ IF STUFF .EQS. "" THEN GOTO END_WRITE     !Test for end of file
$ WRITE IN_FILE STUFF                       !Write to the file
$ GOTO WRITE                                !Goto beginning
$END_WRITE:                                 !End loop
$ !
$ CLOSE IN_FILE                             !Close the file
```

NOTE: The logical name in the OPEN command must be unique. If the OPEN command does not work and your commands seem correct, change the logical name in the OPEN command. Use the SHOW LOGICAL command to display logical name definitions.

If you want to create a file with a unique name, use the F$SEARCH lexical function to see whether the name is already in the directory. (See the lexical function descriptions in the DCL Commands section for more information about F$SEARCH.) The following command procedure prompts the user for a file name, then uses the F$SEARCH lexical function to search the default directory for the name. If a file with that name already exists, control is passed to ERROR_1, the procedure prints the message *File already exists*, and control returns to the label GET_NAME. You are again prompted for a file name.

```
$ ! FILES.COM
$ !
$GET_NAME:
$ INQUIRE FILE "File"          ! Get a file name
$ CHECK = F$SEARCH (FILE)      ! Make sure the file name is unique
$ IF CHECK .NES. "" THEN GOTO ERROR_1
$ OPEN/WRITE IN_FILE 'FILE'  ! Open and write to the file
    .
    .
    .
$ EXIT
$ERROR_1:
$ WRITE SYS$OUTPUT "File already exists"
$ GOTO GET_NAME
```

6.7.3 Reading from a File

To read data from a file, take the following steps:

1. Open the file—The OPEN/READ command opens the file for read access and associates the file name with a logical name.
2. Begin the read loop—File I/O is always done in a loop unless you are reading or writing a single record.
3. Read the data from the file—Use the READ command with the /END_OF_FILE qualifier to read the next record in the file to a symbol. The /END_OF_FILE qualifier causes the VMS system to pass control to the label specified by the /END_OF_FILE qualifier when you reach the end of the file. Generally, you specify the label that marks the end of the read loop.
4. Process the data—When you read a file sequentially, process the current record before reading the next one.
5. Return to the beginning of the loop—You remain in the loop until you reach the end of the file.
6. End the loop and close the file—The CLOSE command disassociates the file name from the logical name and closes the file.

The following command procedure reads and processes each record in the file STAT.DAT:

```
$ OPEN/READ OUT_F STAT.DAT                !Open the file
$ !
$READ_DATA:                               !Begin the loop
$ READ/END_OF_FILE=END_READ OUT_F STUFF !Read a record; test for
$                                         !  end of file
$                                         ! Process the data
    .
    .
    .
$ GOTO READ_DATA                          !Go to the beginning
$                                         ! of the loop
$END_READ:                                !End of loop
$ !
$ CLOSE OUT_F                             !Close the file
```

6.7.4 Modifying a File

You can modify a file in the following ways:

- Rewrite records—This method allows you to make minor changes to a small number of records in a file. You cannot change the size of a record or the number of records in the file.
- Rewrite the file—This method allows you to change, delete, and insert records. You create a new file using the old file as the main source of input.
- Append records to a file—This method allows you to add new records to the end of the file.

6.7.4.1 Minor Modifications

To make minor changes to the records in a file, take the following steps:

1. Open the file for both read and write access.
2. Use the READ command to read through the file until you reach the record that you want to modify.
3. Create a symbol containing the modified record. The modified record must be exactly the same size as the original record. If the text of the modified record is shorter, pad the record with spaces. If the text of the modified record is longer, you cannot use this method to modify the file.
4. Use the WRITE/UPDATE command to write the modified record back to the file.
5. Repeat steps 2 through 4 until you have changed all records you intend to change.
6. Close the file.

Since this method does not allow you to modify the size of the record, use it only if you have formatted the records in a file (for example, in a data file).

The following command procedure reads each record in a data file. The record is displayed on the terminal, and you are asked whether the record is to be modified. If you choose to modify the record, a new record is read from the terminal, and its length is compared to the length of the original record. If the original record is longer, the new record is padded with spaces. If the original record is shorter, an error message is displayed, and you are again prompted for a new record. If you choose not to modify the record, the next record is read from the file.

```
$ ! MODIFY.COM
$ !
$ SPACES = "          "          ! Initialize string of spaces
$                                ! for padding
$ !
$ OPEN/READ/WRITE FILE STATS.DAT ! Open the file
$ !
$BEGIN_LOOP:                     ! Begin the loop
$ !
```

```
$ READ/END_OF_FILE=END_LOOP FILE RECORD ! Read and display a record
$PROMPT:
$ WRITE SYS$OUTPUT RECORD
$ !
$ ! Does the user want to change the record?
$ INQUIRE/NOPUNCTUATION YN "Change? [Y] "
$ IF YN .EQS. "N" THEN GOTO BEGIN_LOOP  ! If not, get next record
$ INQUIRE NEW_RECORD "New record"       ! Otherwise, get the new record
$ !
$ OLD_LEN = F$LENGTH (RECORD)           ! Compare the old and new records
$ IF OLD_LEN .GE. F$LENGTH(NEW_RECORD) THEN GOTO NO_ERROR
$  ! New record longer than old record
$  WRITE SYS$OUTPUT "ERROR -- New record is too long"
$  GOTO PROMPT
$ !
$NO_ERROR:
$ IF OLD_LEN .EQ. F$LENGTH(NEW_RECORD) THEN GOTO WRITE_RECORD
$ ! New record shorter than old record
$ PAD = F$EXTRACT(0,OLD_LEN-F$LENGTH(NEW_RECORD),SPACES)
$ NEW_RECORD = NEW_RECORD + PAD
$ !
$WRITE_RECORD:                          ! Write the new record
$ WRITE/UPDATE FILE NEW_RECORD
$ GOTO BEGIN_LOOP
$ !
$END_LOOP:
$ CLOSE FILE
$ EXIT
```

6.7.4.2 Major Modifications

To make extensive changes to a file, open that file for read access and open a new file for write access. Since the /WRITE qualifier opens a new file for write access, the new file can have the same name as the original file. The new file has a version number one greater than the version number of the old file.

NOTE: You must open the existing file for read access before you open the new version for write access to ensure that the correct file is opened for reading.

To make major modifications to a file, take the following steps:

1. Open the file for read access. This is the file you are modifying.
2. Open a new file for write access.
3. Use the READ command to read each record from the file you are modifying.

   As you read each record from the original file, decide how the record is to be treated. In the following examples, the symbol RECORD contains the record read from the original file:

   - No change—Write the same symbol to the new file.

     ```
     $ ! No change
     $ WRITE NEW_FILE RECORD
     ```

- Change—Use the INQUIRE command to read a different record into the symbol, then write the modified symbol to the new file.

```
$ ! Change
$ INQUIRE NEW_RECORD "New record"
$ WRITE NEW_FILE NEW_RECORD
```

- Delete—Do not write the symbol to the new file.
- Insert—Use a loop to read records into the symbol and to write the symbol to the new file, as shown in the following example:

```
$ ! Insertion
$LOOP:
$ !Get new records to insert
$ INQUIRE NEW_RECORD "New record"
$ IF RECORD .EQS. "" THEN GOTO END_LOOP
$ WRITE NEW_FILE NEW_RECORD
$ GOTO LOOP
$END_LOOP:
```

4. Continue reading and processing records until you have finished.
5. Use the CLOSE command to close both the input and the output files.

6.7.4.3 Appending Records to a File

The OPEN/APPEND command allows you to append records to the end of an existing file. Use the following steps to append records to a file:

1. Use the OPEN command with the /APPEND qualifier to position the record pointer at the end of the file. The /APPEND qualifier does not create a new version of the file.
2. Use the WRITE command to write new data records. Continue adding records until you are through.
3. Use the CLOSE command to close the file.

6.7.5 Handling Input/Output (I/O) Errors

Use the /ERROR qualifier with the OPEN, READ, or WRITE command to suppress error messages and to pass control to a specified label if an error occurs during an input or output operation. This qualifier overrides all other error-control mechanisms (except the /END_OF_FILE qualifier on the READ command). In the following command procedure, if an error occurs during execution of the OPEN command, the message *Error opening STAT.DAT* is printed and the procedure exits:

```
$ OPEN/READ/ERROR=READ_ERR OUT_F STAT.DAT
    .
    .
    .
$ EXIT
$READ_ERR:
$ WRITE SYS$OUTPUT "Error opening STAT.DAT"
$ EXIT
```

6.8 Complex Command Procedures

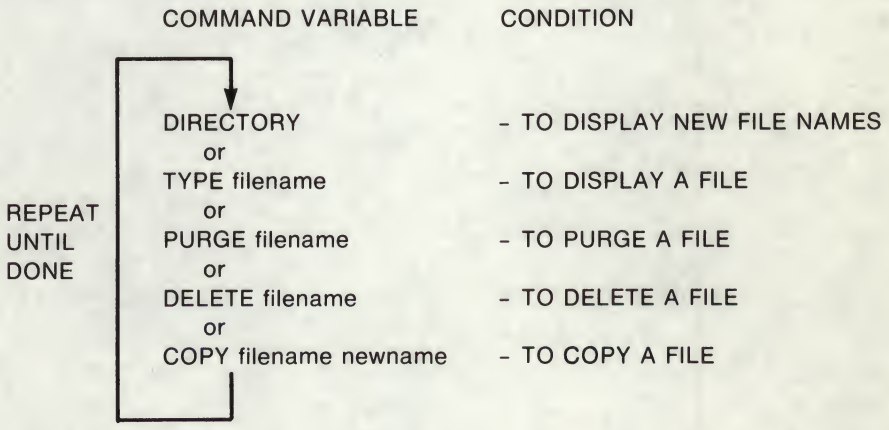
Complex command procedures perform programlike functions. You can use variable input in a complex command procedure, execute sections of the procedure only if certain conditions are true, execute subroutines, or invoke other command procedures. The following sections describe how to design, code, and test complex command procedures.

6.8.1 Designing Complex Command Procedures

Before writing a complex command procedure, perform the tasks interactively that the command procedure will execute. As you type the necessary commands, note the following:

- Variables—Data that changes each time you perform the task.
- Conditionals—Any command or set of commands that may vary each time you perform the task. Note the commands and the conditions under which you would execute them.
- Iteration—Any command or set of commands that you repeat. Note the commands and the factor that controls how often you repeat them.

The following example shows the commands needed to clean up a directory:



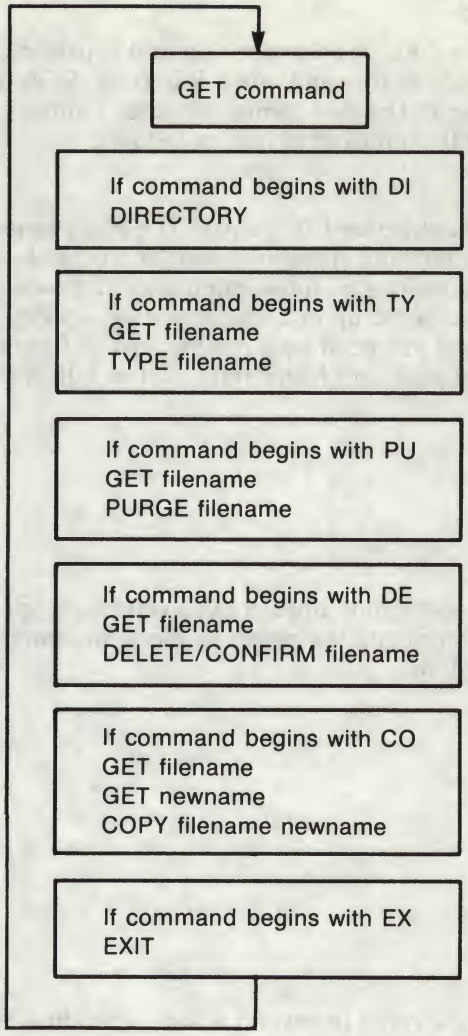
ZK-1750-84

The file names change each time you clean your directory; therefore, they are variables. Any or all of the commands may be executed depending on the operation you need to perform; therefore, each command is conditional. The entire process is repeated until the directory is clean; therefore, it is iterative.

You must decide how to load the variables, test the conditionals, and exit from the loop. For the directory cleaning procedure, the following design decisions were made:

- Load variables—The command procedure gets the file names from the terminal.
- Test conditionals—The command procedure gets a command name from the terminal and executes the appropriate statements based on the command name. The first two characters of each command must be read to differentiate between DELETE and DIRECTORY.
- Exit from loop—You must enter the EXIT command to exit from the loop.

Complete the design as follows:



ZK-1751-84

6.8.2 Coding Complex Command Procedures

To make the command procedure easier to understand and to maintain, try to write the statements so that the procedure executes in a linear fashion, from the first command to the last command. The following sections describe how to execute conditional code and loops. (See Section 5.6.2 for information about the logical operators used in condition expressions.)

6.8.2.1 The IF Command

The IF command tests the value of an expression and causes different commands to execute when the expression is true and when it is false. DCL provides two distinct formats for the IF command. The first format executes a single command when the condition specified to the IF command is true as follows:

```
$ IF condition THEN command
```

DCL also provides a block-structured IF format. The block-structured IF command executes more than one command if the condition is true and accepts an optional ELSE statement that executes one or more commands if the condition is false. To execute more than one command upon a true condition, specify the THEN statement as a verb (a DCL command preceded by a dollar sign) and terminate the resulting block-structured statement with an ENDIF statement as follows:

```
$ IF condition
$ THEN  command
$       command
   .
   .
   .
$ ENDIF
```

To execute one or more commands upon a false condition, specify the ELSE statement as a verb and terminate the resulting block-structured statement with an ENDIF statement as follows:

```
$ IF condition
$ THEN  command
$       command
   .
   .
   .
$ ELSE  command
$       command
   .
   .
   .
$ ENDIF
```

Command blocks can be executed in several ways, depending on whether you leave the commands in the same command procedure or put them in another command procedure and execute them there:

- If you leave the commands in the command procedure, place them after the THEN statement.

  ```
  $ IF condition
  $ THEN  command
          command
     .
     .
     .
  $ ENDIF
  ```

- If you place the commands in a separate procedure, make the call to that command procedure as part of the THEN statement.

```
$ IF condition
$ THEN @command_procedure
$ ELSE command
$      command
$ ENDIF
```

You can continue to specify the nonblock structured IF format and direct flow to a labeled region when the condition specified is met as follows:

```
$ IF not condition THEN GOTO END_LABEL
    .
    .
    .
$END_LABEL:
```

In the following example, a specified file is purged if COMMAND equals "PU." If COMMAND does not equal "PU", a specified file is printed.

```
$! Purge a file.  If no file exists, print the requested file.
$ IF COMMAND .EQS. "PU"
$ THEN
$    INQUIRE FILESPEC "File to purge"
$    PURGE 'FILESPEC'
$ ELSE
$    INQUIRE FILESPEC "File to print"
$    PRINT 'FILESPEC'
$ ENDIF
$!Type a file.  If no file exists, exit"
$ IF COMMAND .EQS. "TY"
    .
    .
    .
$ EXIT
```

In the following example, the command procedure SCREEN_SETUP.COM is executed if F$MODE() equals "INTERACTIVE". If F$MODE() does not equal "INTERACTIVE", the procedure exits. (The command procedure SCREEN_SETUP.COM would be in a separate file; the commands it contains are indented here for clarity.)

```
$ IF F$MODE() .EQS. "INTERACTIVE"
$ THEN
$    @SCREEN_SETUP
         $ ! SCREEN_SETUP.COM
         $ ! Set terminal characteristics
         $ SET TERMINAL/DEVICE=VT200
         $ SET TERMINAL/WIDTH=132
$ ! Invoke Editor
$    EVE :== EDIT/TPU
$ ELSE
$    EXIT
$ ENDIF
    .
    .
    .
```

6.8.2.2 Case Statements

A case statement is a special form of conditional code that executes one out of a set of command blocks, depending on the value of a variable or expression. Typically, the valid values for the case statement are labels at the beginning of each command block. The case statement passes control to the appropriate block of code by using the specified value as the target label in a GOTO statement.

To write a case statement:

1. List the labels—Equate a symbol to a string that contains a list of the labels delimited by slashes (or any character you choose to act as a delimiter). This symbol definition should precede the command blocks.

   ```
   $ COMMAND_LIST = "/PURGE/DELETE/EXIT/"
   ```

2. Write the "case statement"—First, use the INQUIRE command to get the value of the case variable. Next, use the IF command with F$LOCATE and F$LENGTH to determine whether the value of the case variable is valid. If the case variable is valid, execute the case statement (a GOTO command) to pass control to the appropriate block of code. Otherwise, display a message and exit or request a different case value.

   In the following example, the label is equated to the full command name. Therefore, F$LOCATE includes the delimiters in its search for the command name to ensure that the command is not abbreviated.

   ```
   $GET_COMMAND:
   $ INQUIRE COMMAND -
     "Command (EXIT,PURGE,DELETE)"
   $ IF F$LOCATE ("/"+COMMAND+"/",COMMAND_LIST) .EQ. -
      F$LENGTH (COMMAND_LIST) THEN GOTO ERROR_1
   $ GOTO 'COMMAND'
      .
      .
      .
   $ERROR_1:
   $ WRITE SYS$OUTPUT "No such command as ''COMMAND'"
   $ GOTO GET_COMMAND
   ```

3. Write the command blocks—Each block of commands may contain one or more commands. Begin each command block with a unique label. End each command block by passing control to a label outside the list of command blocks.

```
$GET_COMMAND:
    .
    .
    .
$PURGE:
$ INQUIRE FILE
$ PURGE 'FILE'
$ GOTO GET_COMMAND
$ !
$DELETE:
$ INQUIRE FILE
$ DELETE 'FILE'
$ GOTO GET_COMMAND
$ !
$EXIT:
```

6.8.2.3 Loops

A loop is a group of commands that executes repeatedly until a condition is met. The following arrangement is recommended for statements that form a loop:

1. Begin the loop.
2. Change the termination variable.
3. Test the termination variable. If the condition is met, go to the end of the loop.
4. Perform the commands in the body of the loop.
5. Return to the beginning of the loop.
6. End the loop.

You can also write loops that test the termination variable at the end of the loop rather than at the beginning as follows:

1. Begin the loop.
2. Perform the commands in the body of the loop.
3. Change the termination variable.
4. Test the termination variable. If the condition is not met, go to the beginning of the loop.
5. End the loop.

Note that when you test the termination variable at the end of the loop, the commands in the body of the loop execute at least once, regardless of the value in the termination variable.

Both of the following examples execute a loop that terminates when COMMAND equals "EX" (EXIT). (F$EXTRACT truncates COMMAND to its first two characters.) In the first example, COMMAND, the termination variable, is tested at the beginning of the loop; in the second, it is tested at the end of the loop.

```
$ ! EXAMPLE 1
$ !
$GET_COMMAND:
$ INQUIRE COMMAND-
  "Command (EXIT,DIRECTORY,TYPE,PURGE,DELETE,COPY)"
$ COMMAND = F$EXTRACT(0,2,COMMAND)
$ IF COMMAND .EQS. "EX" THEN GOTO END_LOOP
   .
   .
   .
$ GOTO GET_COMMAND
$END_LOOP:
```

```
$ ! EXAMPLE 2
$ !
$GET_COMMAND:
$ INQUIRE COMMAND-
  "Command (EXIT,DIRECTORY,TYPE,PURGE,DELETE,COPY)"
$ COMMAND = F$EXTRACT(0,2,COMMAND)
   .
   .
   .
$ IF COMMAND .NES. "EX" THEN GOTO GET_COMMAND
$ ! End of loop
```

To perform a loop a specific number of times, use a counter as the termination variable. In the following example, 10 file names are input by the user and placed into the local symbols FIL1, FIL2, . . . , FIL10:

```
$ NUM = 1                        ! Set counter
$LOOP:                           ! Begin loop
$ INQUIRE FIL'NUM' "File"        ! Get file name
$ NUM = NUM + 1                  ! Update counter
$ IF NUM .LT. 11 THEN GOTO LOOP  ! Test for termination
$END_LOOP:                       ! End loop
   .
   .
   .
```

To perform a loop for a known sequence of values, use F$ELEMENT. In the following example, the files CHAP1, CHAP2, CHAP3, CHAPA, CHAPB, and CHAPC are processed in order:

```
$ FILE_LIST = "1,2,3,A,B,C"
$ INDEX = 0
$PROCESS:
$ NUM = F$ELEMENT(INDEX,",",FILE_LIST)
$ IF NUM .EQS. "," THEN GOTO END_LOOP
$ FILE = "CHAP''NUM'"
$ ! process file named by FILE
   .
   .
   .
$ INDEX = INDEX + 1
$ GOTO PROCESS
$END_LOOP:
$ EXIT
```

6.8.2.4 Subroutines

Use the GOSUB command or the CALL command to transfer control to a subroutine within a command procedure. The GOSUB command transfers control to a labeled subroutine in a command procedure without creating a new procedure level. Since the GOSUB command does not create a new command level, it is referred to as a *local* subroutine call. The RETURN command terminates the GOSUB subroutine procedure, returning control to the command following the calling GOSUB statement.

The following command procedure shows how to use the GOSUB command to transfer control to labeled subroutines:

```
$!
$! GOSUB.COM
$!
$ SHOW TIME
$ GOSUB TEST1
$ WRITE SYS$OUTPUT "success completion"
$ EXIT
$!
$! TEST1 GOSUB definition
$!
$ TEST1:
$     WRITE SYS$OUTPUT "This is GOSUB level 1."
$     GOSUB TEST2
$     RETURN
$!
$! TEST2 GOSUB definition
$!
$ TEST2:
$     WRITE SYS$OUTPUT "This is GOSUB level 2."
$     RETURN
```

The CALL command transfers control to a labeled subroutine in a command procedure and creates a new command level. The CALL command allows you to keep more than one related command procedure in a single file, making the procedures easier to manage. You can pass up to eight parameters to the subroutine; you can create up to 32 command levels with the CALL command.

By default, the CALL command sends output to SYS$OUTPUT. The optional /OUTPUT qualifier allows you to direct output from the subroutine to a file. Do not use wildcard characters in the output file specification. The default file type for the output file is LIS.

Unless they are masked using the SET SYMBOL command, local symbols defined in an outer level are available to any inner procedure or subroutine levels and global symbols are available at any command level. Labels are valid only for the level in which they are defined.

The SUBROUTINE and ENDSUBROUTINE commands define the beginning and end of a subroutine invoked with the CALL command. The label defining the entry point to the subroutine immediately precedes the SUBROUTINE command. The ENDSUBROUTINE command functions as an EXIT command if an EXIT command is not specified in the procedure. The ENDSUBROUTINE command terminates the

subroutine and transfers control to the command line immediately following the CALL command.

Command lines in a CALL subroutine execute only when the subroutine is called with the CALL command. During the line-by-line execution of the command procedure, the command language interpreter skips all commands between the SUBROUTINE and the ENDSUBROUTINE commands.

The following procedure shows how to use CALL to transfer control to a labeled subroutine. The example also shows that you can call another command procedure from within a subroutine. (You can also call another subroutine from a subroutine.) The CALL command invokes the subroutine SUB1, directing output to the file NAMES.LOG and allowing other users write access to the file.

```
$!
$! CALL.COM
$!
$! Define subroutine SUB1
$!
$ SUB1: SUBROUTINE
   .
   .
   .
$ @FILE           !Invoke another command procedure
   .
   .
   .
$ EXIT
$ ENDSUBROUTINE   !End of SUB1 definition
$!
$! Start of main routine. At this point, SUB1 has
$! been defined, but none of the commands in the
$! subroutine have executed.
$!
$ START:
$  CALL/OUTPUT=NAMES.LOG SUB1 "THIS IS P1"
   .
   .
   .
$  EXIT !Exit this command procedure file
```

6.8.3 Testing and Debugging

For lengthy or complex command procedures, write the logic for the main procedure, but use stubs for the nested procedures and subroutine-type pieces of code. A stub is a command that writes a message stating the function it is replacing. For example, the following stub replaces the purge routine:

```
   .
   .
   .
$ ! Purge a file
$ IF COMMAND .NES. "PU" THEN GOTO END_PURGE
$ WRITE SYS$OUTPUT "Purge routine"   ! stub
$END_PURGE:
   .
   .
   .
```

If you have a number of places that need stubs, you can use one nested command procedure to insert the stub logic as follows. (The command procedure STUB.COM would be in a separate file; it is indented here for clarity.)

```
   .
   .
   .
$ ! Purge a file
$ IF COMMAND .NES. "PU" THEN GOTO END_PURGE
$ @STUB "Purge"
$        ! STUB.COM
$        ! Procedure STUB
$         WRITE SYS$OUTPUT "''P1' routine"
$END_PURGE:
   .
   .
   .
```

Once you have written the code using stubs, you can test the overall logic of the command procedure as follows. Test all possible paths of execution.

```
$ ! CLEAN.COM
$ !
$GET_COMMAND:
$ ! Read a command from the terminal
$ INQUIRE COMMAND-
  "Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY)"
$ COMMAND = F$EXTRACT(0,2,COMMAND)
$ !
$ IF COMMAND .EQS. "EX" THEN GOTO END_COMMAND
$ !
$ ! Purge a file
$ IF COMMAND .NES. "PU" THEN GOTO END_PURGE
$ WRITE SYS$OUTPUT "Purge routine."
$END_PURGE:
$ !
$ ! Delete a file
$ IF COMMAND .NES. "DE" THEN GOTO END_COMMAND
$ WRITE SYS$OUTPUT "Delete routine."
$ END_DELETE:
   .
   .
   .
$ GOTO GET_COMMAND
$END_COMMAND:
```

Once the overall logic of the procedure works, you can begin filling in the stubs. Fill in the first stub, test it, and debug it if necessary. When that stub works, move on to the next one.

The following commands are useful for debugging command procedures:

- SET VERIFY—SET VERIFY prints each line before it is executed. When an error occurs with verification set, you see the error and the line that generated the error. In the following example, seeing the command line that generated the error explains the error message.

```
$ SET VERIFY
$ @CDIR
$ ! Read a command from the terminal
$ INQUIRE COMMAND-
  "Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY)"
Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY):  DELETE
$ COMMAND = F$EXTRACT(0,2,COMMAND)
$GET_COMMAND:
$ IF COMMAND .EQ. "EX" THEN GOTO END_COMMAND
%DCL-W-IVCHAR, non-numeric character in value string
 \E\ EXIT
$ IF COMMAND .NES. "DI" THEN GOTO END_DIR
    .
    .
    .
```

  The logical operator .EQ. is used to compare numbers, not strings (see Section 5.6.2 for information about logical operators). To correct the error, change .EQ. to .EQS.. Note that you can use keywords with SET VERIFY to indicate that only command lines or data lines are to be verified.

- SHOW SYMBOL—Use the SHOW SYMBOL command to print the values of the symbols involved in an error. In the following procedure, the IF statements are not passing control to the expected procedures. Putting the command SHOW SYMBOL COMMAND before the IF statements allows you to check the value of COMMAND.

```
$ SET VERIFY
$ @CDIR
$GET_COMMAND:
$ ! Read a command from the terminal
$ INQUIRE COMMAND-
  "Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY)"
Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY):  DELETE
$ COMMAND = F$EXTRACT(1,2,COMMAND)
$ SHOW SYMBOL COMMAND
  COMMAND = "EL"
$GET_COMMAND:
$ IF COMMAND .EQS. "EX" THEN GOTO END_COMMAND
    .
    .
    .
```

The F$EXTRACT lexical function is extracting two characters beginning at character 1 (the second character) rather than at character 0 (the first character). To correct the error, change F$EXTRACT(1,2,COMMAND) to F$EXTRACT(0,2,COMMAND). Note that INQUIRE automatically converts input to uppercase; therefore, the quoted string in the IF statement must be written in uppercase for DCL to evaluate the strings as equal.

6.9 Handling Errors and CTRL/Y Interrupts

The following table describes the default action taken when an error occurs or when you press CTRL/Y. These default actions can be overridden with the ON, SET [NO]ON, and SET [NO]CONTROL=Y commands.

Interrupt	Default Action
Error or severe error	Procedure exits to the next command level.
CTRL/Y at DCL command level or command level 1	Interrupts procedure: procedure can continue if no other image forces it to exit.
CTRL/Y at command level lower than level 1	Procedure exits to the next higher command level.

6.9.1 The ON Command

The ON command specifies an action to be performed if an error of a certain severity or greater severity occurs. (See Section 5.2 for discussion of error conditions and severity levels.) When such an error occurs, the system takes the following actions:

1. The error message is displayed.
2. The action specified by the ON command is performed.
3. The default error action (exit to the next higher command level) is reset.

When an error of less than the specified severity occurs, the error message is displayed, and the command procedure continues executing. Assume a command procedure executes the following command:

```
$ ON ERROR THEN GOTO ERR1
```

The command procedure continues to execute unless an error or severe error occurs. When such an error occurs, the error message is displayed; the default error action (exit to the next higher command level) is reset; and the command procedure continues execution at ERR1. If a second error occurs before another ON or SET NOON command is executed, the procedure exits to the next higher level.

The action specified by the ON command applies only within the command level in which the command is executed. Therefore, if you execute an ON command in a procedure that calls another procedure, the ON command action does not apply to the nested command procedure.

The execution of an ON statement performs an implicit SET ON function, thus nullifying any SET NOON condition that may be in effect.

NOTE: Only one ON statement of each type can be in effect at any one time. For example, if the command procedure includes more than one ON ERROR statement, the ON ERROR statement executed most recently is in effect.

6.9.2 The SET [NO]ON Command

The SET ON and SET NOON commands enable and disable error checking for the current command level. The SET NOON command overrides the ON command. If an error (regardless of severity) occurs after a SET NOON command is executed, the system takes the following actions:

1. Displays the error message
2. Continues executing the procedure

SET NOON remains in effect until either an ON or SET ON command is executed. The SET ON command reenables error checking for the current command level. The action specified by the last ON command, if one exists, is reestablished. Otherwise, the default error action is reenabled.

In the following command procedure, if an error or severe error occurs while copying the file, the procedure continues to execute without going to GET_COMMAND as specified by the ON command.

```
$ ON ERROR THEN GOTO GET_COMMAND
$GET_COMMAND:
    .
    .
    .
$ ! Type a file
$ IF COMMAND .NES. "CO" THEN GOTO END_COPY
$ INQUIRE FILESPEC "File to move"
$ INQUIRE COPYSPEC "New file specification"
$ SET NOON
$ COPY 'FILESPEC' 'COPYSPEC'
$ SET ON
$END_COPY:
    .
    .
    .
```

6.9.3 CTRL/Y Interrupts

The ON CONTROL_Y command specifies an action to be performed when CTRL/Y is pressed at the current command level. (By default, pressing CTRL/Y causes the system to prompt for command input at the CTRL/Y command level.) In the following command procedure, pressing CTRL/Y while a file is being typed passes control to the label END_TYPE.

```
    .
    .
    .
$ ! Type a file
$ IF COMMAND .NES. "TY" THEN GOTO END_TYPE
$ ON CONTROL_Y THEN GOTO END_TYPE
$ TYPE 'FILESPEC'
$END_TYPE:
$ !
$ !Reset default
$ SET NOCONTROL=Y
$ SET CONTROL=Y
    .
    .
    .
```

An ON CONTROL_Y command remains in effect until another ON CONTROL_Y or a SET NOCONTROL=Y command executes or the command procedure exits.

See Section 6.11 for another example of using the ON CONTROL_Y command.

To exit from a nonterminating loop when CTRL/Y is disabled, you must delete your process from another terminal using the DCL command STOP. If you disable the default CTRL/Y action, reset it as soon as possible. To reset the default CTRL/Y action, execute the SET NOCONTROL=Y command followed by the SET CONTROL=Y command, as shown in the previous example.

6.10 Restarting Batch Jobs

Chapter 3 describes how to specify that your batch job be reexecuted if the system crashes before the job is finished. By default, a batch job is reexecuted beginning with the first line. However, you can use the following symbols in your command procedures to specify a different restarting point:

- $RESTART—A global symbol whose value is true if the batch job has been started at least once before this execution. Do not specify a value for $RESTART; the system will assign the appropriate value.
- BATCH$RESTART—A global symbol whose value you specify using the SET RESTART_VALUE command.

The following steps describe how to use these symbols in a command procedure:

- Begin each possible starting point of the procedure with a label.

- As the first step in each section, equate the value of BATCH$RESTART to the label using the SET RESTART_VALUE command.
- At the beginning of the procedure, test $RESTART. If $RESTART is true, issue a GOTO statement using BATCH$RESTART as the transfer label.

The following command procedure extracts a number of modules from a library, concatenates those modules, and then sorts the resulting file. If aborted, the command procedure reexecutes from the beginning of the file, the statement labeled CONCATENATE_LIBRARIES or the statement labeled SORT_FILE, depending on the value of BATCH$RESTART. (If you were extracting a number of separate modules, you could make each extraction a separate section.)

```
$ ! SORT_MODULES.COM
$ !
$ ! set default to the directory containing
$ ! the library whose modules are to be sorted
$ SET DEFAULT WORKDISK:[ACCOUNTS.DATA83]
$
$ ! check for restarting
$ IF $RESTART THEN GOTO BATCH$RESTART
$
$ EXTRACT_LIBRARIES:
$ SET RESTART_VALUE=EXTRACT_LIBRARIES
            .
            .
            .
$ CONCATENATE_LIBRARIES:
$ SET RESTART_VALUE=CONCATENATE_LIBRARIES
            .
            .
            .
$ SORT_FILE:
$ SET RESTART_VALUE=SORT_FILE
            .
            .
            .
$ EXIT
```

6.11 Cleanup Operations

In general, execution of a command procedure should not change the user's process state. Therefore, a command procedure should include a set of commands that returns the process to its original state. Common cleanup operations include the following (see the lexical function descriptions in the DCL Commands section for lexical function specifications):

- Closing files—If you have opened any files, make sure that they are closed before the command procedure exits. You can use the lexical function F$GETJPI to examine the remaining open file quota (FILCNT) for the process. If FILCNT is the same at the beginning and end of the command procedure, you know that no files have been left open. In the following example, a warning message is displayed if a file is left open:

```
$ FIL_COUNT = F$GETJPI("","FILCNT")
    .
    .
    .
$ IF FIL_COUNT .NE. F$GETJPI("","FILCNT") THEN-
  WRITE SYS$OUTPUT "WARNING -- file left open"
```

- Deleting temporary or extraneous files—If you have created temporary files, delete them. In general, if you have updated any files, you should purge them to delete the previous copies. Take care in deleting files that you have not created. For example, if you have updated a file that contains crucial data, you may wish to make the purging operation optional.
- Resetting default device and directory—If you change the default device and/or directory, reset the original defaults before the command procedure exits.

  To save the name of the original default directory, use the DEFAULT keyword of the F$ENVIRONMENT lexical function. At the end of the command procedure, include a SET DEFAULT command that restores the saved device and directory.

  The following example saves and restores device and directory defaults:

```
$ SAV_DEFAULT = F$ENVIRONMENT("DEFAULT")
    .
    .
    .
$ SET DEFAULT 'SAV_DEFAULT'
```

The following table lists other commonly changed process characteristics as well as the lexical functions and commands used to save and restore the original settings:

Characteristic	To Save . . .	To Restore . . .
DCL prompt	F$ENVIRONMENT	SET PROMPT
Default protection	F$ENVIRONMENT	SET PROTECTION/DEFAULT
Privileges	F$SETPRV	F$SETPRV or SET PROCESS/PRIVILEGES
Control characters	F$ENVIRONMENT	SET CONTROL
Verification	F$VERIFY	F$VERIFY
Message format	F$ENVIRONMENT	SET MESSAGE
Key state	F$ENVIRONMENT	SET KEY

To ensure that cleanup operations are performed even if the command procedure is aborted, begin each command level in the command procedure with the following statement:

```
$ ON CONTROL_Y THEN GOTO CLEAN_UP
```

In each command level of the command procedure, place cleanup operations after the CLEAN_UP label.

Chapter 7

Maintaining Accounts and System Security

By being aware of system security, you can take steps to protect your files from unauthorized access. This chapter discusses security features of the VMS operating system and describes how to protect your files and use the system securely.

7.1 User Accounts

User accounts are maintained in a file named SYS$SYSTEM:SYSUAF.DAT, referred to as the user authorization file, or UAF. A VMS system uses the UAF to validate login requests and to set up processes for users who successfully log in. You can examine and modify this file with the Authorize Utility (AUTHORIZE).

The UAF contains a record for each account. Each record consists of fields providing information for the following areas: identification, login characteristics, login restrictions, priority, limits, and privileges. The user name field is specified as a parameter to Authorize Utility commands; the other fields are specified as qualifier values of Authorize Utility commands.

7.2 Protection

The VMS operating system provides two related mechanisms to control the access that users have to system objects as follows:

- UIC-based protection—Each user process in the system is assigned a user identification code (UIC) in the user authorization file (UAF) with the Authorize Utility. Each object on the system, such as a file, is also assigned a UIC (typically the UIC of its creator). Each object also maintains a protection mask, a structure which defines the type of access allowed to users, based upon the relationship between the user UIC and the object UIC.

- ACL-based protection—An access control list (ACL) specifying the type of access to be granted or denied to a particular user or group of users can be associated with a system object. An ACL is an optional form of protection that is typically created by the object owner using the ACL editor (invoked with the DCL command EDIT/ACL) or the SET ACL command.

The system objects for which ACL-based protection can be specified are: files, directories, devices, batch and print queues, logical name tables, and global sections. Users are specified by identifiers in the rights database that are assigned with the Authorize Utility. (The rights database is described in Section 7.2.2.2.)

The system determines whether to grant a user access to an object, as follows:

1. ACL—If the user matches an identifier in the object's ACL, the system grants or denies access based on the ACL. However, even if an entry in the ACL denies access, the system may still grant access based on the SYSTEM and OWNER fields of the UIC-based protection (see Section 7.2.1.2).
2. UIC—If the user does not match an identifier in the object's ACL or the object has no ACL, the system grants or denies access based on the relationship between the user's UIC and the object's UIC as specified in the object's protection mask.
3. Privileges—If the system denies the user access, the user may be granted access by using one of the following privileges: BYPASS, GRPPRV, READALL, or SYSPRV. Privileges are described in the *VMS System Manager's Manual.*

UIC-based protection is useful for denying or granting access to a specified group of users or to all users on the system. The optional ACL-based protection allows further control over the protection of an object. You can grant or deny access to individual users, and you can further identify users by certain aspects of their usage (such as whether they are interactive, batch, local, remote, or dialup users). The combination of UIC and ACL protection provides a way to specify multiple subsets and overlapping groups of users.

7.2.1 UIC-Based Protection

Typically, you use UIC-based protection if the object is to be accessed by: (1) only the owner, (2) all users on the system, or (3) a specific group of users.

7.2.1.1 UIC Format

UICs are a subset of the identifiers that the VMS system uses to identify users and groups of users (see Section 7.2.2.2 for other identifiers). Each user on the system is assigned a unique UIC when the account is created. A UIC consists of two parts, group and member, specified in the following format:

[group,member]

A UIC can be either numeric or alphanumeric.

- Numeric UIC—Consists of a group number in the range 0 through 37776 (octal) and a member number in the range 0 through 177776 (octal).

- Alphanumeric UIC—Consists of a member name (the user name parameter specified with the Authorize Utility command ADD) and, optionally, a group name. The UIC group name is taken from the account name specified with the /ACCOUNT qualifier. Member and group names must contain at least one alphabetic character and up to a maximum of 31 alphanumeric characters (including A through Z, 0 through 9, underscore, and dollar sign characters). An alphanumeric UIC is equated to a numeric UIC in the rights database by default once the rights database has been created. You can generally specify a numeric UIC and its equivalent alphanumeric UIC interchangeably.

The member component of a UIC must be unique to the system. By default, the member component of an alphanumeric UIC is equated to both the group and member components of a numeric UIC in the rights database (so that specifying just the member part of an alphanumeric UIC is sufficient). The following examples illustrate several UICs in proper UIC notation:

UIC	Meaning
[200,10]	Group 200, member 10
[3777,3777]	Group 3777, member 3777
[USER,FRED]	Group USER, member FRED
[EXEC,JONES]	Group EXEC, member JONES
[JONES]	Group EXEC, member JONES

When you log in to a VMS system, the UIC of your process is the UIC specified in your UAF account. Typically, your process UIC does not change, although it can be changed with the SET UIC command (which requires CMKRNL privilege). By default, detached processes (created by the DCL command SUBMIT or RUN) and subprocesses (created by the DCL command SPAWN) take the same UICs as their creators. If you have DETACH privilege, you can create a detached process with a different UIC (by using the /UIC qualifier of the RUN command).

By default, an object (such as a file) receives the UIC of the process creating it. You can change the UIC of a file with the /OWNER_UIC qualifier of the BACKUP, CREATE, SET DIRECTORY, and SET FILE commands. With SYSPRV privilege, you can specify any UIC; with GRPPRV privilege, you can specify any member within your current group; otherwise, you can specify only your own UIC.

You can specify the UIC of a disk volume with the /OWNER_UIC qualifier of the INITIALIZE and MOUNT commands (except for the system disk, which retains the UIC specified when it was initialized). You can also change the UIC of a disk (including the system disk) with the SET VOLUME command. You must have the VOLPRO privilege to specify a UIC other than your own. In addition, when you initialize a system disk (/SYSTEM qualifier), it receives a UIC of [1,1] and a group disk (/GROUP qualifier) receives a UIC of [n,0], where *n* is the group number of the owner. You can specify a UIC for a device such as a terminal (by default, devices are

not owned) with the /OWNER_UIC qualifier of the SET PROTECTION/DEVICE command.

7.2.1.2 Ownership and Access Categories

The relationships between the UIC of a process and the UIC of an object fall into the following four ownership categories:

- System—The UIC of the process is in the range 1 through 10 (octal) or the process has SYSPRV privilege. (The range of system UICs is determined by the SYSGEN parameter MAXSYSGROUP, which defaults to 10 octal.)
- Owner—The UIC of the process and the UIC of the object are identical.
- Group—The group number of the process and the group number of the object are identical or the process has GRPPRV privilege.
- World—All users on the system.

A process may be able to access an object through more than one of these ownership categories. For example, a user with a UIC of [CS102,MARTIN] can attempt access to an object with a UIC of [CS102,PROF] through both the group and world categories.

A process can access an object in the following ways:

- Read (allocate)—Read a file; read from a disk volume; allocate nonfile devices.
- Write—Write a file; write to a disk volume.
- Execute (create)—Execute an image file; look up entries in a directory if you explicitly specify the file name (without using wildcard characters); create files on a disk volume.
- Delete—Delete files.

7.2.1.3 Protection Masks

A protection mask is a structure created by the system for each system object defining the type of UIC access allowed to the object. A protection mask consists of four fields, each with four indicators. Each field applies to one category of ownership. Each indicator within a field applies to one category of access. The fields and indicators are as follows:

Ownership Fields	Access Indicators			
SYSTEM	READ	WRITE	EXECUTE	DELETE
OWNER	READ	WRITE	EXECUTE	DELETE
GROUP	READ	WRITE	EXECUTE	DELETE
WORLD	READ	WRITE	EXECUTE	DELETE

Protection for an object must be specified in the following format:

(ownership[:access],...)

Specify ownership as one of the following (each may be abbreviated to one character): SYSTEM, OWNER, GROUP, or WORLD. Specify access as one or more of the following indicators: R (read), W (write), E (execute), D (delete). Omission of the colon and access indicators disallows access for that category of ownership. The following protection specification allows system users full access to an object, the owner full access except delete, and group and world users no access:

```
(S:RWED,O:RWE,G,W)
```

Omission of an ownership category generally results in no access being granted to that category. However, the SET PROTECTION command retains the existing protection of the object for omitted fields of the ownership category.

7.2.1.4 Securing User Data and Devices

The suggestions for establishing UIC-based protection of data and devices belonging to individual and application accounts are as follows:

- Default protection—Make sure your default protection is adequate. In general, you do not want to grant write or delete access to world users. You may or may not want to grant write access to group users. You may or may not want to grant read access to world users.

  By default, the system assigns each file the following protection mask:

  ```
  (S:RWED,O:RWED,G:RE,W)
  ```

  You can redefine this default for all files you create using the SET PROTECTION /DEFAULT command, or you can modify the protection on individual files using the /PROTECTION qualifier to the SET FILE command.

- Sensitive files—Protect sensitive files by specifying extra protection, for example, with the SET PROTECTION command. You can also protect sensitive files by maintaining them in a subdirectory on which extra protection is set. However, to protect sensitive files completely, directory protection alone is not adequate. You must also protect each individual file contained within the directory. You can add a default protection ACE to the subdirectory file that defines the UIC protection mask for newly created files in the directories (see Section 7.2.2.5).

- Individual access—Use access control lists (ACLs) to grant or deny individual users access to a file (see Section 7.2.2).

- Copied files—In general, do not copy a file into someone else's directory (for example, if you have write access to a group member's directory), as it will have your UIC instead of the UIC of the directory's owner. Use the MAIL command to send the file, or have the owner of the directory copy the file.

- Private volumes—A private volume is one that is mounted on a device allocated to your process. No other users on the system can access the volume. If you mount a private volume, use the DEALLOCATE command to deallocate the device when you are done.

7.2.2 ACL-Based Protection

Typically, you use access control lists (ACLs) on system objects to grant or deny access to individual users, groups of users, and to subsets of user groups. ACLs contain entries (ACEs) that specify the access to be granted or denied a user or group of users. The user or group of users is designated by identifiers, as described in Section 7.2.2.2.

7.2.2.1 Object Types

You can establish ACLs for various system objects: files, directory files, devices, global sections, queues, and logical name tables. In general, you need not be concerned about the object type when establishing or changing an ACL; however, ACLs set up on devices, logical name tables, and global sections (except those backed by files) are not saved and must be reestablished every time the system is booted.

ACLs on system objects are set up and modified with the ACL editor or with the DCL command SET ACL in the following format:

SET ACL/OBJECT_TYPE=object-type object-name ...

For example, the following command adds an ACL to the file PERSONNEL.DAT containing a single ACE that denies all network access to the file:

```
$ SET ACL/OBJECT_TYPE=FILE PERSONNEL.DAT -
_$  /ACL=(IDENTIFIER=NETWORK,ACCESS=NONE)
```

7.2.2.2 Identifiers

An identifier is a value that represents an individual user, or a group of users, for an aspect of the user's environment. Following are the three types of identifiers:

- UIC identifiers
- General identifiers
- System-defined identifiers

A UIC identifier is created each time a new user account is added with the Authorize Utility. The UIC identifier matches the UIC specified for the user. A second UIC identifier is created matching the name of the UIC group when a UIC group is defined for the first time. UICs can be in both numeric and alphanumeric form (see Section 7.2.1.1) and are useful in identifying individual users in ACLs. UIC identifiers must be enclosed in brackets and can have wildcard characters in either the group or member fields (for example, [EXEC,*]).

Identifiers include other general identifiers that you explicitly associate with users in the rights database. These general identifiers are useful in identifying multiple groups of users outside the bounds of UIC groups. For example, you could create the identifier SECRET and assign it in the rights database to a selected group of users, some of whom could be in different UIC groups.

A third type of identifier includes the following system-defined identifiers, which you can use to identify users by their mode of using the system:

System-Defined Identifiers	Type of User
BATCH	Batch user
DIALUP	User logged in on a dialup terminal
INTERACTIVE	Interactive user
LOCAL	User logged in on local terminal
NETWORK	Network process
REMOTE	User logged in over the network

Generally, you should treat the preceding system-defined identifiers as being mutually exclusive. However, you can combine them with UIC identifiers or general identifiers by connecting them with plus signs (for example, [FRED]+BATCH). Access is granted only if both identifiers are true. (In the example [FRED]+BATCH, the user identified as [FRED] must be running a batch job for the system to grant the specified access.)

Following are examples of user-defined identifiers that are valid for ACL-based protection:

- PAYROLL—Specifies all users holding the identifier PAYROLL.
- [USER,JONES]—Specifies the user whose alphanumeric UIC is group USER and member JONES.
- [200,10]—Specifies the user whose numeric UIC is group 200, member 10.
- [FRED]+BATCH—Specifies the batch user whose alphanumeric UIC is FRED.
- DIALUP—Specifies all users logged in on a dialup terminal.

Rights List

The system determines protection by checking identifiers in the object's ACL against the list of identifiers held by the accessor to find a matching entry. The list of identifiers held by the accessor is called a *rights list*.

The file containing all the identifiers defined in the system is called the *rights database*. (Identifiers are defined and granted to specific users with the AUTHORIZE commands ADD/IDENTIFIER and GRANT/IDENTIFIER.) The rights list, created for each process at login, is the portion of the rights database containing all the identifiers and attributes held by the user.

7.2.2.3 Access Control List Entries (ACEs)

An entry in an access control list specifies the access to a system object that is to be granted or denied a user. This access is specified by the identifier. Different kinds of access are NONE, READ, WRITE, EXECUTE, DELETE, and CONTROL.

Following are the three types of ACEs

- Identifier ACE
- Default_protection ACE
- Alarm_Journal ACE

7.2.2.4 IDENTIFIER ACEs

An identifier ACE controls the type of access allowed to a particular user or group of users. Identifier ACEs have the following format:

(IDENTIFIER=identifier[,OPTIONS=options+...],ACCESS=access+...)

The following are examples of ACEs:

```
(IDENTIFIER=[200,201],ACCESS=READ+WRITE+EXECUTE)
```

Grants the user identified by UIC identifier [200,201] read, write, and execute access to the system object.

```
(IDENTIFIER=[FRED]+BATCH,ACCESS=WRITE+EXECUTE)
```

Grants batch user with the alphanumeric UIC [FRED] write and execute access to the system object.

```
(IDENTIFIER=PAYROLL,ACCESS=READ)
```

Grants users who hold the identifier PAYROLL read access to the system object.

```
(IDENTIFIER=DIALUP,ACCESS=NONE)
```

Denies holders of the system-defined identifier DIALUP any access to the system object.

The preceding ACEs could be specified in a single ACL for a system object as follows:

```
(IDENTIFIER=[200,201],ACCESS=READ+WRITE+EXECUTE)
(IDENTIFIER=[FRED]+BATCH,ACCESS=WRITE+EXECUTE)
(IDENTIFIER=PAYROLL,ACCESS=READ)
(IDENTIFIER=DIALUP,ACCESS=NONE)
```

To specify one or more default ACEs for inclusion in the ACLs of files subsequently created in a directory, use the OPTIONS=DEFAULT option of an identifier ACE. The following ACE, when placed in the ACL of the directory file, grants all users holding the SECRET identifier read, write, and execute access to new files in the directory:

```
(IDENTIFIER=SECRET,OPTIONS=DEFAULT,ACCESS=READ+WRITE+EXECUTE)
```

NOTE: Default protection is associated only with newly created files, not existing ones. If you add a default protection ACE to a directory, you must also change the protection on files already in the directory.

Because the system determines access at the first matching entry, the order of the entries is critical. For example, if the last entry in the previous example—(IDENTIFIER=DIALUP,ACCESS=NONE)—were placed at the top of the list, all dialup users (including those specified in the remaining entries) would be denied access to the associated file. Placing it last in the access control list allows users holding identifiers [200,201], [FRED]+BATCH, and [PAYROLL] the specified access, even when they are dialup users.

7.2.2.5 DEFAULT_PROTECTION ACEs

To specify default protection for new files in a particular directory, place a default_protection ACE in the ACL of the directory file. (The directory file is the file with the directory name as file name and DIR as file type in the parent directory.) The default_protection ACE affects files that are subsequently created in the directory and in any subdirectories under that directory unless protection is specified for one of those files individually. Default_protection ACEs apply UIC-based protection. Specify a default_protection ACE in the following format, where *protection-mask* is the same mask used in UIC protection (see Section 7.2.1.3):

```
(DEFAULT_PROTECTION[,options],protection-mask)
```

The following default_protection ACE specifies that by default the system and owner have read, write, execute, and delete access to any files subsequently created for the directory and that group and world users have no access.

```
(DEFAULT_PROTECTION,S:RWED,O:RWED,G,W)
```

7.2.2.6 ALARM_JOURNAL ACEs

The alarm_journal ACE allows you to specify that an alarm message be sent to the security operator's terminal if a certain type of access takes place. Alarms are supported only for files and global sections. The alarm_journal ACE functions only when alarms to the security operator's terminal have been enabled through the DCL command SET AUDIT, security messages to the operator's terminal have been enabled with the DCL command REPLY/ENABLE=SECURITY, and the OPCOM process is executing.

Following is the format of an alarm_journal ACE:

```
(ALARM_JOURNAL=SECURITY[,options+...][,access+...])
```

The following alarm_journal ACE specifies that an alarm will be sent to the security operator's terminal if an accessor attempts to read the object, and that this ACE will be preserved even when an attempt is made to delete the entire ACL:

```
(ALARM_JOURNAL=SECURITY,OPTIONS=PROTECTED,ACCESS=READ+SUCCESS+FAILURE)
```

For an alarm to have any effect, you must include either SUCCESS or FAILURE or both in the ACCESS field.

7.2.3 File Protection

File protection is usually transparent. To set protection or modify the ACL of a file, you must own the file, have control access to the file, or have GRPPRV, SYSPRV, BYPASS, or READALL privilege.

NOTE: To completely protect a file, you must apply the same or greater protection to the directory in which the file resides. See Section 7.2.3.3 for information on directory protection.

7.2.3.1 Default File Protection

A new file receives default UIC-based protection and the default ACEs (if any) of its parent directory. A renamed file's protection is unchanged. A new version of an existing file receives the UIC-based protection and ACL of the previous version. (Use the /PROTECTION qualifier of the BACKUP, COPY, CREATE, and SET FILE commands to override the default UIC-based protection.)

You can use either of the following methods to override the default UIC-based protection given to new files:

- Default UIC protection—The operating system provides each process with a default UIC-based protection of (S:RWED,O:RWED,G:RE,W). This indicates that SYSTEM users and the owners of objects have full access to the object, users in the same UIC group as the object owner have read and execute access to the object, and all other users are denied access to the object. To change the default protection, invoke the SET PROTECTION command with the /DEFAULT qualifier. For example, if you place the following command in your login command procedure, you grant all processes read and execute access to any files that you create. (Remember that you must execute the login command procedure for this command to execute.)

  ```
  $ SET PROTECTION = (S:RWED,O:RWED,G:RE,W:RE)/DEFAULT
  ```

- Default ACL protection—You can override default UIC protection for specified directories or subdirectories by placing a default_protection ACE in the ACL of the appropriate directory file. The default protection specified in the ACE is applied to any new file created in the specified directory or subdirectory of the directory. The following ACE, which must be in the ACL of a directory file, specifies that the default protection for that directory and the directory's subdirectories allow system and owner processes full access, group processes read and execute access, and world users no access.

```
(DEFAULT_PROTECTION,S:RWED,O:RWED,G:RE,W:)
```

To specify a default identifier ACE to be copied to the ACL of any file subsequently created in the directory, specify the DEFAULT option in the directory file's identifier ACL.

7.2.3.2 Explicit File Protection

You can explicitly specify UIC-based protection for a new file with the /PROTECTION qualifier (valid with the BACKUP, COPY, and CREATE commands) as shown in the following example:

```
$ CREATE MAST12.TXT/PROTECTION=(S:RWED,O:RWED,G,W)
```

You can change the UIC-based protection on an existing file with the SET PROTECTION command as shown in the following example:

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W) MAST12.TXT
```

After a file is created and you have created an ACL for the file, you can modify the ACL and add as many ACEs to the ACL as you want. The protection specified by the ACL overrides the file's UIC protection.

7.2.3.3 Directory Protection

You cannot completely protect a file without applying at least the same protection to the directory in which the file resides. For example, if you deny a user all access to a file but allow that user read access to the file's directory, the user cannot access the contents of the file but can see that it exists. Conversely, a user allowed access to a file and denied access to the file's directory (or one of the parent directories) cannot see that the file exists.

NOTE: To protect sensitive files, directory protection alone is not adequate. You must also protect each file within the directory.

By default, top level directories receive UIC-based protection (S:RWE,O:RWE,G:RE,W:E) and no ACL. Subdirectories receive UIC-based protection from the parent directory.

To specify UIC-based protection explicitly when creating a directory, use the /PROTECTION qualifier of the CREATE/DIRECTORY command. You cannot specify an ACL for the directory until the directory is created. To change the UIC-based protection of an existing directory, use the SET PROTECTION command (apply this command to the directory file). To specify or change the ACL of an existing directory, edit the directory file's ACL (see Section 7.2.3.1).

You can limit but not prohibit directory access by specifying execute access but not read access. Execute access on a directory permits you to examine and read files that you know are contained in the directory (that is, you know the file specifications), but prevents you from displaying a list of the files in the directory.

7.2.3.4 Mail File Protection

Mail files receive the protection (S:RWD,O:RW,G,W). Files of type MAI created with the EXTRACT command of the Mail Utility receive the protection (S:RWD,O:RWD,G,W).

Chapter 8

Editing Files with the EVE and EDT Editors

Text editors are computer programs that allow you to enter text from a keyboard into computer memory. Once the text is in memory, you can modify the text using text editing commands. For example, you can type in data for a report and then rearrange sections, duplicate information, or substitute phrases. Text editors also format text so that the report or article can be ready for distribution. Different formatting commands set margins, insert white space, and paste in figures and tables.

VMS supports several text editors. This chapter discusses the EVE and the EDT editors.

8.1 The EVE Editor

EVE, the Extensible VAX Editor, is an editor built on the VAX Text Processing Utility (VAXTPU), a high performance, programmable, text processing utility. Using EVE, you can create and edit new files or edit existing files. You can add text to a file and modify or format that text. EVE is interactive, so you see the changes to a file as you make them.

Unlike the EDT editor, EVE lets you display more than one buffer on the screen at a time and to edit more than one file during the same editing session. EVE is easy to customize or extend using EVE commands and VAXTPU procedures.

8.1.1 Beginning and Ending an Editing Session

To begin an editing session, invoke EVE with the DCL command EDIT/TPU. In an editing session, you can create and edit a new file, or you can edit an existing file. The session ends when you enter the EXIT or QUIT command. Exiting from EVE typically produces a new file or a new version of an existing file.

8.1.1.1 Invoking EVE

You can start an editing session by creating a new file and inserting text into it during the session. You can also begin by specifying an existing file when you invoke EVE.

To begin an EVE editing session, enter the DCL command EDIT/TPU. Specify a file name on the command line if you want to edit an existing file or assign a name to a new file.

For example, to invoke EVE to create a new file named NEWFILE.DAT, enter the following command:

```
$ EDIT/TPU MYFILE.DAT
```

This command produces the following screen output:

```
[End of file]


Buffer NEWFILE.DAT                                  | Insert | Forward

Editing new file: could not find WORKDISK:[USER]NEWFILE.DAT
```

EVE inserts the text of the file you are editing into a temporary holding area called a *buffer*. A buffer is a temporary storage area and exists only during an editing session. The contents of the buffer are shown in an area of your screen that is called a *window*. When you end an editing session, you direct EVE to save or discard the contents of a buffer.

The end-of-file marker defines the end of an EVE buffer. It is only visible on the screen and does not become part of your file.

A highlighted status line appears at the bottom of the EVE window and provides information about the EVE buffer. The status line shows the buffer name, current mode (insert or overstrike), and current direction (forward or reverse).

If you invoke EVE with a file name, an informational message appears in the message buffer beneath the highlighted status line stating either that the file is a new file or that a certain number of lines were read from an existing file. EVE communicates with you throughout the editing session by displaying messages in the Message window.

To invoke EVE to edit an existing file named OLDFILE.DAT, enter the EDIT/TPU command in the following format:

```
$ EDIT/TPU OLDFILE.DAT

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer OLDFILE.DAT                                    | Insert | Forward

4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Rather than name a file at the beginning of an editing session, you can invoke EVE with the command EDIT/TPU and then enter text into the buffer. You can save the text by writing it to a file using the WRITE FILE command described in Section 8.1.7. Alternately, when you finish creating your file, EVE prompts for a file name as follows:

```
Enter a file name to write buffer MAIN; else press RETURN:
```

Type the name of the file, and press RETURN to write out the buffer to a file.

8.1.1.2 Ending an Editing Session

Two different commands can end an EVE editing session. EVE produces a new version of the edited file when you end the session with the EXIT command. EVE discards your edits when you end a session with the QUIT command. Any existing versions of the files remain unchanged regardless of how the editing session is ended.

To save your edited text, use the EXIT command. Enter the EXIT command by pressing the F10 key (on VT200-series or VT300-series terminals) or by pressing CTRL/Z.

If you have modified the current buffer, EVE creates a new version of the file with the same file name and file type as the original version, with the version number incremented by 1. For example, if you use the EXIT command after modifying a file named FUN.DAT;1, the output file is named FUN.DAT;2, as follows:

```
Buffer FUN.DAT                                        | Insert | Forward

4 lines written to file WORKDISK:[USER]FUN.DAT;2
```

To exit from a session without saving your edits, use the QUIT command. To execute the QUIT command, press the DO key (PF4 on VT100-series terminals), type QUIT at the *Command:* prompt, and press RETURN. For example, if you have modified a file named FUN.DAT and enter the QUIT command, the following display appears on your terminal screen:

```
Buffer FUN.DAT                                        | Insert | Forward

Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press RETURN if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press RETURN, and exit from the file using the EXIT command.

If you have modified buffers other than the current one, EVE asks if you want to save the contents of those buffers. If you type Y, EVE creates new versions of any existing files, incrementing the version number by 1. EVE prompts for a file name if no file currently exists.

8.1.2 Entering EVE Commands

Once you have invoked EVE, enter EVE commands to edit and manipulate text. There are two ways to enter EVE commands: by pressing predefined editing keys and by typing the actual commands.

8.1.2.1 Using Defined Keys to Enter EVE Commands

EVE defines some editing keys by default. You can define additional editing keys to enter commands or to perform editing operations that you use frequently (see Section 8.1.9). The predefined editing keys on VT200-series and VT300-series terminals include the minikeypad (located between the main keypad and the numeric keypad), certain function keys, and certain control key sequences. On VT100-series terminals, EVE automatically defines most of the numeric keypad keys, the arrow keys, and certain control keys. Each predefined editing key performs one editing command.

Throughout this chapter, EVE editing keys are referred to by their names, rather than by their location on the VT200-series, VT300-series, or VT100-series keyboards. Figure 8–1 shows the predefined keys on the VT200-series and VT300-series terminal. Figure 8–2 shows the predefined keys on the VT100-series terminal.

Figure 8–1: Editing Keys—VT200-Series and VT300-Series Terminals

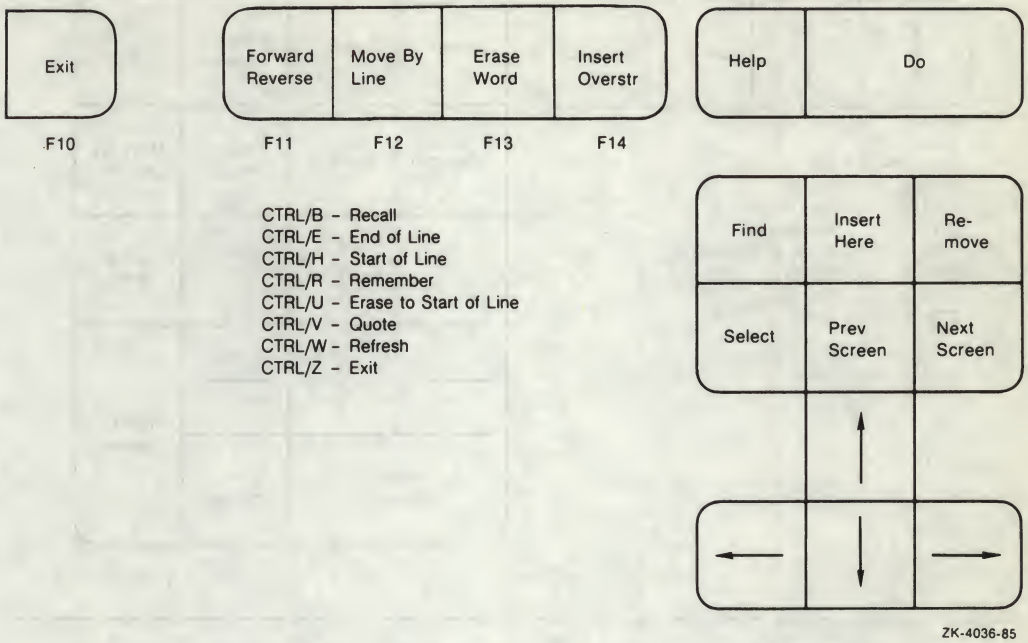
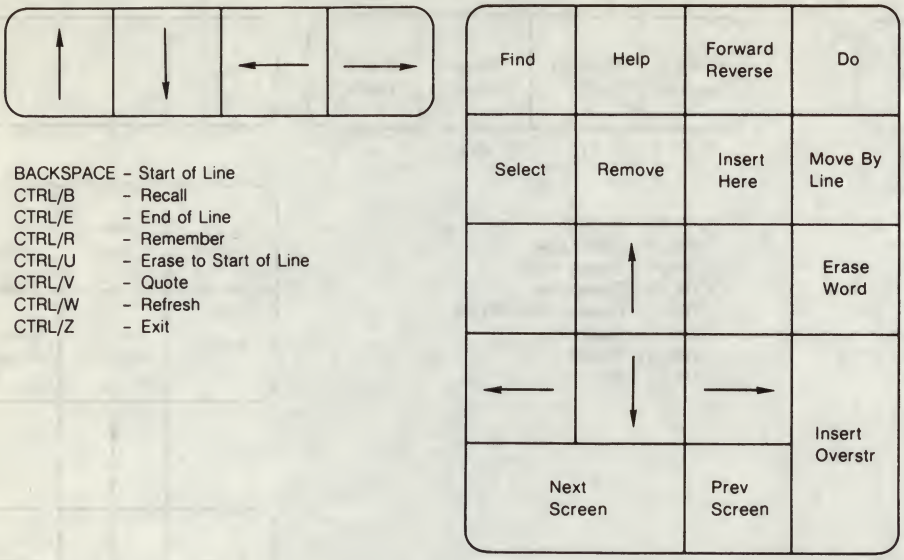


Figure 8-2: Editing Keys—VT100-Series Terminals



ZK-4037-85

EVE offers two default keypads in addition to the default EVE keypad. You can select an EDT keypad or a WPS keypad. Although neither fully implements EDT or WPS editing functions, each provides most keypad functions.

8.1.2.2 Entering EVE Commands

In addition to using defined keys, you can enter EVE commands by typing them at the *Command:* prompt. When you enter EVE commands, you always perform the following three steps:

1. Press the DO key. EVE displays the *Command:* prompt.
2. Type the EVE command after the prompt.
3. Press either RETURN or the DO key to enter the command.

You can correct typing mistakes on the EVE command line by pressing DCL line-editing keys. Use CTRL/U to erase to the beginning of the line, CTRL/E to move to the end of the line, and CTRL/B to recall the last command entered. By default, the editing mode of the EVE command line is the same as the editing mode of your terminal. (You can change the default prior to invoking EVE with the DCL command SET TERMINAL. Once in EVE, you can change the editing mode with CTRL/A.)

To save keystrokes when typing EVE commands, you can use CTRL/B, abbreviate commands, use the REPEAT command, or press the DO key. Each method of saving keystrokes is described as follows:

- Press CTRL/B to recall the last EVE command you entered. Continue pressing CTRL/B until the command you wish to execute appears on your screen, and press RETURN to enter the command.
- Abbreviate EVE command names, making sure the abbreviation is unambiguous.
- Use the REPEAT command to repeat an EVE command or keystroke. For example, to insert the character *p* in your editing buffer 20 times, press the DO key, type REPEAT 20, and press RETURN. Then type *p*. EVE inserts a *p* into the current buffer 20 times.
- Press the DO key twice to activate the last command entered.

8.1.3 Editing Text

Once you know how to invoke the EVE editor and how to enter commands, you can use EVE commands to edit new and existing files. Editing keys and commands allows you to position the cursor and perform such text editing operations as moving, erasing, and restoring text.

8.1.3.1 Moving the Cursor

When editing files with EVE, first you move the cursor to the place in the text where you want to perform an editing function. Therefore, the more quickly and efficiently you move the cursor through the text, the more time you save in your editing session.

The following tables show the EVE editing keys and commands that move the cursor:

Editing Key	Moves the Cursor to the Following Position
Up arrow	Up one line.
Down arrow	Down one line.
Left arrow	One character or column to the left.
Right arrow	One character or column to the right.
CTRL/E	End of the current line.
CTRL/H	Beginning of the current line.
Move By Line	In forward direction: end of the current line or if the cursor is already at the end of a line, to the end of the next line. In reverse direction: beginning of the current line or if the cursor is already at the beginning of a line, to the beginning of the previous line.
Next Screen	Forward in the current buffer. The number of lines the cursor moves depends on the size of the current window.
Previous Screen	Backward in the current buffer. The number of lines the cursor moves depends on the size of the current window.

Editing Command	Moves the Cursor to the Following Position
BOTTOM	End of the current buffer.
BUFFER	Puts the specified buffer in the current window and moves the cursor to the end of the buffer. Creates a new buffer if the specified buffer does not exist.
END OF LINE	End of the current line.
GET FILE	Creates a new buffer containing text of the specified file (or an empty buffer if the file does not exist) and puts the cursor at the beginning of the buffer.
LINE	Beginning of the specified line in the current buffer.
MOVE BY PAGE	Next or previous page break, depending on the current direction.
MOVE BY WORD	Beginning of the next word, if direction is forward. In reverse direction, cursor moves to the beginning of the current word; if already there, EVE positions cursor at the beginning of the previous word.
NEXT WINDOW	Next window on your screen, assuming another exists. The cursor appears in the last location it occupied in that editing window.
PREVIOUS WINDOW	Previous window on screen, assuming another window exists. The cursor appears in the last location it occupied in that editing window.
SET CURSOR BOUND	Changes cursor mode. Cursor follows the flow of text and cannot be put into an unused portion of the buffer.
SET CURSOR FREE	The default mode. Cursor is not bound to the flow of the text but can be put anywhere on the screen and text can be entered.
START OF LINE	Beginning of the current line.
TOP	Beginning of the current buffer.

8.1.3.2 Inserting Text

You can insert sections of text, entire files, and special nonprinting characters (such as control characters) into the buffer you are currently editing. The following tables show the editing keys and EVE commands that you use while inserting text:

Editing Key	What It Does
Insert Overstrike	Changes the current editing mode as displayed on the highlighted status line. In insert mode, EVE inserts text at the current character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position.
CTRL/A	Same as the Insert Overstrike key.
CTRL/V	Lets you insert nonprinting characters (or control codes) in a buffer. You can search for special characters using the Find key. First, press the Find key, then press CTRL/V and the special character to be found, and activate the search by pressing RETURN.

Command	What It Does
INCLUDE FILE	Inserts the entire contents of the specified file into a buffer at the line before the current cursor location.

Press CTRL/A or the Insert Overstrike key to change from one mode to the other.

You can add text to your buffer in the following ways:

- Text — Type characters that EVE adds to the buffer at the current cursor position. The characters are added according to the current mode of the buffer (insert or overstrike).
- Files — Add entire files by pressing the DO key and entering the EVE command INCLUDE FILE. EVE disregards the current mode (insert or overstrike) of the buffer and inserts the entire contents of the specified file into the buffer just before the line in which the cursor currently appears.
- Special Nonprinting Characters — Add special nonprinting characters by pressing CTRL/V followed by the special character.

8.1.3.3 Erasing and Restoring Text

With the EVE editor, you can easily delete text from a file or correct mistakes made during an editing session. If you erase text by mistake, you can restore the most recently erased text to its former location or, by moving the cursor, to another location. The following table shows the editing keys and EVE commands that erase and restore text:

Editing Key	What It Erases
<X	Character to the left of the cursor.
Erase Word	Current word or, if the cursor is not on a word, erases the next word.
CTRL/U	All characters from the current cursor position to the beginning of the line.

Command	What It Does
ERASE CHARACTER	Erases the current character.
ERASE LINE	Erases from the current cursor position to the end of the current line, appending the next line to the end of the current line.
ERASE PREVIOUS WORD	Erases the previous word or the word the text cursor is on.
RESTORE	Restores, at the current cursor position, the word, sentence, or line that you have erased most recently with an EVE command or editing key.
RESTORE CHARACTER	Restores, at the current cursor position, the character you have erased most recently with an EVE command or editing key.

Command	What It Does
RESTORE LINE	Restores, at the current cursor position, the line that you have erased most recently with an EVE command or editing key.
RESTORE WORD	Restores, at the current cursor position, the word that you have erased most recently with an EVE command or editing key.

To erase text from your buffer, move the cursor to the location of the text that you want to erase, and press the appropriate editing key, or type the appropriate EVE command.

8.1.3.4 Moving Text from One Location to Another

The following table describes the functions of the Select, Remove, and Insert Here keys as well as the STORE TEXT command, which are used to erase text, to move text from one location to another within a buffer in "cut and paste" operations, and to duplicate text. For information on how to move text from one buffer to another, see Section 8.1.7.

Editing Key	What It Does
Select	Marks text (highlighting it in reverse video) from the cursor location to wherever you move the cursor. To cancel the selection, press the Select key again or use RESET.
Remove	Removes the text that was marked with SELECT, and places it in the Insert Here buffer.
Insert Here	Inserts the text from the Insert Here buffer at the current cursor location.

Command	What it Does
STORE TEXT	Copies text that was marked with SELECT or highlighted by FIND, placing it in the Insert Here buffer. Text that is copied is not removed from its original position.

To mark text when the buffer is set in a forward direction, place the cursor on the first character that you wish to erase. Press the Select key, and then move the cursor to one character beyond the last character that you wish to erase. (In reverse direction, move the cursor to the last character, not one beyond.) The text that will be erased is highlighted in reverse video. (If you decide not to remove text from the buffer, press the Select key again to cancel the selection.) Press the Remove key. EVE deletes the highlighted text from your screen and places it in the Insert Here buffer.

You can insert the text at any cursor location by pressing the Insert Here key, or you can erase text permanently from your buffer by leaving it in the Insert Here buffer.

The following example shows how to erase and move text from one location to another using the Select, Remove, and Insert Here keys. Invoke EVE to edit the file RHYMES.DAT.

Move the cursor to the beginning of the second line of RHYMES.DAT and press the Select key.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]


Buffer RHYMES.DAT                                    | Insert | Forward

Move the cursor to select text.
```

Press the Down Arrow key once. The second line of text is highlighted. Press the Remove key. The second line of text is removed from the current buffer.

```
She rhymes with tree,
and this one makes three.
[End of file]


Buffer RHYMES.DAT                                    | Insert | Forward

Remove completed.
```

Press the Down Arrow key once. Press RETURN twice and then press the Insert Here key. The text in the Insert Here buffer is inserted at the current cursor location.

```
She rhymes with tree,
and this one makes three.


also with bee,
[End of file]


Buffer RHYMES.DAT                                    | Insert | Forward
```

8.1.3.5 Locating Text

Use the Find key to locate specified strings of text in your buffer. Press the Find key (PF1 on VT100-series terminals). Then type the text string that you wish to locate, called the *search string*, and press RETURN.

EVE attempts to move the cursor to the beginning of the specified string.

If the search string contains all lowercase letters, EVE disregards the case of letters and locates any occurrence of the string. Thus, *the*, *THe*, and *thE* all match the search string *the*. If the search string contains one or more uppercase letters, EVE locates only the occurrences of the string in which the case of letters is exactly the same. Therefore, the only match for the search string *tHis* is tHis.

The current direction of the buffer determines whether EVE first searches in a forward or a reverse direction.

If the editor cannot find the string in the current direction but finds it in the opposite direction, EVE prompts you to change direction. To search in the opposite direction, type *Y*. EVE moves the cursor to the first occurrence of the string in the opposite direction. The current direction in the highlighted status line is not changed, however.

When EVE finds the search string, the editor highlights it and moves the cursor to the first letter of the string.

If you press the Find key twice, EVE tries to find the next occurrence of the search string.

The following example uses the existing file RHYMES.DAT to illustrate the use of the Find key. When you invoke EVE to edit RHYMES.DAT, the cursor appears on the first letter of the first line of the buffer, and the current direction is forward.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the Find key, type the letters *ree*, and press RETURN. The cursor moves to the letter *r* in the word tree and highlights the letters *ree*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
Forward Find:  ree
```

8.1.3.6 Marking Locations in Text

The MARK and GO TO commands are very useful when you are editing a large file and know that you want to return to a specific cursor location later in the editing session. The following table describes the MARK and GO TO commands:

Command	What It Does
MARK	Associates a unique and invisible label, consisting of one or more alphanumeric characters, with the current cursor location. The mark exists for the rest of an editing session.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer and places the buffer in the current window.

To mark a cursor location, press the DO key, type MARK label-name, and press RETURN. The label name can be one or more printable characters, including

alphanumeric and punctuation characters. To return the cursor to the marked location, press the DO key, type GO TO label-name, and press RETURN.

8.1.3.7 Replacing Text

The REPLACE command allows you to replace a text string that appears in the current buffer with another text string. This is especially useful if you have spelled a word incorrectly throughout a long file, and you want to fix every occurrence.

To use the REPLACE command, press the DO key, type REPLACE, and press RETURN. Type the string that you wish to replace at the *Old string:* prompt and press RETURN. Type the new string at the *New string:* prompt and press RETURN.

If EVE finds the old string in the current direction, it moves the cursor to the first occurrence of the old string, highlights the string, and provides the following prompt: *Replace? Type yes, no, all, last, or quit.*

If EVE does not find the string in the current direction but finds it in the opposite direction, EVE provides the following prompt: *Found in 'reverse/forward' direction. Go there? [Y].* If you type Y, EVE moves the cursor to the first occurrence of the string in the new current direction, highlights the string, and provides the following prompt: *Replace? Type yes, no, all, last, or quit:* The current direction of the buffer in the highlighted status line is not changed.

Respond to the prompt by typing a single-character abbreviation of the response and pressing RETURN. Simply pressing RETURN is equivalent to typing Y and pressing RETURN.

The REPLACE command is case sensitive. If the old string and the new string are given in all lowercase characters, then EVE matches the case appropriately for each replacement. For example, in replacing the string *parsley* with the string *dill*, EVE replaces a capitalized version of parsley with a capitalized version of dill. If the old string is lowercase, the search is general. However, if the old string is uppercase or mixed case, the search is case-exact. If the old and the new strings are lowercase, the replacement mirrors the case of the occurrence. Whereas, if the new string is uppercase or mixed case, the replacement is exact.

The following example shows how to use the REPLACE command to replace every occurrence of the string *ee* with the string *oo*. Move the cursor to the top of the buffer. Press the DO key, type REPLACE, and press RETURN. Type *ee* at the highlighted *Old string:* prompt, press RETURN, and type *oo* at the highlighted *New string:* prompt.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Old string:  ee
```

The cursor moves to the highlighted string *ee* in the word tree. Type *all*, and press RETURN. All occurrences of the string *ee* are replaced with the string *oo*.

```
She rhymes with troo,
also with boo,
and this one makes throo.
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
Replace? Type yes, no, all, last, or quit:  all
```

8.1.4 Using the HELP Facility

EVE has an online HELP Facility that quickly supplies information on editing commands and keys during your editing session without disturbing your work. You can obtain help by entering the HELP command or by pressing the Help key.

To view a list of EVE commands, press the DO key and enter HELP. Use the Previous Screen and Next Screen keys (up and down arrow keys on a VT100-series terminal) to scroll through the entire list of EVE commands. To get information on a particular command, type a command name after the help prompt and press RETURN. The help text appears on the screen.

If you know the name of a specific command for which you need help, press the DO key, type HELP followed by the name of the command, and press RETURN. The help text for that command appears on the screen.

The HELP Facility also provides information on general topics. For example, if you choose to use the EDT or the WPS keypad, use the command HELP SET KEYPAD EDT or the command HELP SET KEYPAD WPS to get information on changing your default keypad. To display a list of all defined keys for the keypad you are using, enter the command HELP KEYS. There is also help on the difference between an EDT keypad within EVE and the original EDT keypad (HELP EDT DIFFERENCES).

The Help key produces a keypad diagram for the keypad you are using. The diagram shows both the default editing keys and the keys you have defined for the minikeypad (LK201 keyboard), the main keypad, the keys F10-F14 (LK201 keyboard), and the GOLD key (described in Section 8.1.9.3).

You can get help on particular editing keys after you display the keypad by pressing the desired key. If you press a key to which you have assigned an EVE command, EVE provides the help text for that EVE command.

8.1.5 Recovering from System Interruptions

EVE has recovery procedures for two types of system interruptions. You can remove extraneous characters that appear on your screen. You can also recover edits from an interrupted editing session with the journaling facility.

8.1.5.1 Refreshing the Screen

If extraneous characters, such as a message from the operator, appear on your terminal screen while you are editing or inserting text, press CTRL/W to refresh your screen. The screen becomes blank, and then all characters are redrawn, minus any extraneous characters.

8.1.5.2 Using the Journal File

If you are editing a file and a system interruption (that is, a break in communication between your terminal and the computer) occurs, you can recover your lost editing session. By default, EVE records the keystrokes you enter during an editing session in a journal file that has the same file name as the file you are editing and a file type of TJL.

Typically, an editing session ends without interruption, so the system deletes the journal file. When you experience a system interruption, however, the journal file is saved. EVE can use the journal file to reconstruct your editing session so that only the last few keystrokes of your editing session are lost.

To recover an editing session, enter the command you used to invoke EVE with the /RECOVER qualifier. For example, to recover an editing session you began with the command EDIT/TPU LETTER.RNO, type the following command and file name and press RETURN:

```
$ EDIT/TPU/RECOVER LETTER.RNO
```

You must recover an editing session at a terminal of the same type as the one you used for your editing session. When EVE finishes recovering the session, check to ensure that the last few keystrokes of your editing session were recovered and continue editing the file. If another system interruption occurs before you exit, a journal file containing the keystrokes from both editing sessions is saved.

The journaling facility has the following restrictions:

- If you use the WRITE FILE command during your editing session to copy the contents of the buffer to another file, you need to recover the original version of the file that you were editing.
- EVE is usually unable to recover keystrokes entered after you press CTRL/C. If you press CTRL/C, end the editing session immediately with the EXIT command and invoke EVE again.

8.1.6 Formatting Text

EVE provides commands that enable you to format your text by setting margins, tabs, screen width, and word wrap. It allows you to center lines, take extra white space out of text, and insert page breaks. The following table lists text formatting commands and describes their functions:

Command	What It Does
CAPITALIZE WORD	Capitalizes a single word or each word in the text highlighted by FIND or SELECT.
CENTER LINE	Centers the line of text marked by the cursor between the current left and right margins.
FILL	Reformats the current paragraph or selected range according to the margins of the buffer, so the maximum number of words fits on a line.
FILL PARAGRAPH	Reformats the paragraph the text cursor identifies according to the margins set for the buffer.
FILL RANGE	Reformats the currently selected range of text (or the current FIND range) according to the current margin settings.
INSERT PAGE BREAK	Inserts a form feed character at the current editing position to mark the beginning of a new page. A page break appears as a small double-F and is always on a line by itself.
LOWERCASE WORD	Makes a single word or the text highlighted with FIND or SELECT all lowercase.
SET LEFT MARGIN	Sets the left margin in current buffer. The left margin must be greater than 0 but less than the right margin. By default, the left margin is 1.
SET RIGHT MARGIN	Sets the right margin for the current buffer. The right margin must be greater than the left margin. By default, the right margin is one less than the screen width. The width is typically 80, so the default margin is typically 79.
SET TABS AT	Sets tab stops at the columns that you specify. By default, tab stops are set in every eighth column.
SET TABS EVERY	Sets tab stops at the specified interval. By default, tab stops are set in every eighth column.
SET TABS INSERT	Turns on tab insert mode, so that EVE sets a tab stop at the column where you press the Tab key, moving over text currently on the screen. SET TABS INSERT is the default mode.
SET TABS SPACES	Changes tab mode to insert an appropriate number of spaces rather than a tab character when the Tab key is pressed. Previously existing tab characters are not affected.
SET TABS MOVEMENT	Changes the tab mode so the Tab key becomes a cursor-control key. Pressing the Tab key moves the cursor to the next tab stop but does not insert a tab character.
SET TABS VISIBLE	Displays tabs as visible characters on the screen.
SET TABS INVISIBLE	Does not display tabs as visible characters on the screen. SET TABS INVISIBLE is the default mode.

Command	What It Does
SET WIDTH	Sets the width of lines displayed on the screen. Specify width as a positive integer *n*. By default, screen width is your terminal setting, typically 80 columns. Setting the width changes the display of text in all windows.
SET WRAP	Enables word wrapping at the right margin of the buffer so EVE starts new lines without a RETURN command or use of the FILL command. SET WORD WRAP is the default setting.
SET NOWRAP	Disables word wrapping at the right margin of the buffer. You must start new lines by pressing RETURN or by using the FILL command.
SHIFT LEFT	Moves the current window to the left a specified number of columns. The SHIFT LEFT command can be used only to reverse the effect of the SHIFT RIGHT command.
SHIFT RIGHT	Moves the current window to the right a specified number of columns, allowing you to view columns of characters that do not currently appear on the terminal screen.
UPPERCASE WORD	Makes a single word or the text highlighted with FIND or SELECT all uppercase.

The following example shows how to use EVE commands to set margins. Invoke EVE to edit the existing file RHYMES.DAT. Press the DO key, type SET LEFT MARGIN 20, and press RETURN to set a left margin of 20. The text that currently appears in the buffer does not change.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
Command: SET LEFT MARGIN 20
```

Move the cursor to the end of the buffer and type the following new text: *Also with thee, and me.* The new text that you enter is inserted at the left margin of 20.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and
                    me.
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
Left margin set to 20
```

8.1.7 Using Buffers

Buffers are storage areas that exist only during an editing session. The following table describes EVE commands used to create, manipulate, and delete buffers:

Command	What It Does
BUFFER	Puts the specified buffer in the current window and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
DELETE BUFFER	Deletes the buffer you specify.
GET FILE	Creates a new buffer that contains the text of the specified file (or an empty buffer if you specify a file that does not exist); places the new buffer in the current window; and places the cursor at the beginning of the new buffer.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer and places the buffer with the label in the current window.
SHOW	Displays information about the buffers you have created during the editing session. If more than one buffer is active in your editing session, EVE displays information about the buffer you are currently editing. For information on other buffers, press the DO key. To resume editing, press any other key.
SHOW BUFFERS	Lists the buffers you have created during an editing session. You can move the cursor through the list and specify a particular buffer for viewing using the Select key.
SHOW SYSTEM BUFFERS	Lists the system buffers created by EVE. You can move the cursor through the list and specify a buffer for viewing using the Select key.
WRITE FILE	Writes the contents of the current buffer to a file. If you do not specify a file name, EVE uses the buffer name as the file name. If you created the current buffer with the BUFFER command, EVE prompts you for a file specification.

When you invoke EVE to edit an existing file, EVE reads the contents of the file into a buffer. The highlighted status line contains the buffer name, editing mode, and current direction of the buffer.

To delete a buffer, enter the DELETE BUFFER command, specifying the name of the buffer you wish to delete. For example, the command DELETE BUFFER MYFILE.TXT deletes the buffer called MYFILE.TXT. The buffer name must be typed in full; no abbreviations are allowed.

8.1.7.1 Listing Buffers

To display a list of all buffers you have created during an editing session, enter the SHOW BUFFERS command. You can scroll through the list and specify a buffer you want to view or any buffers you want to delete. To display a buffer in your current window, move the cursor to the buffer name and press the Select key. To delete a buffer, move the cursor to the buffer name and press the Remove key.

To display a list of all buffers that EVE has created, enter the SHOW SYSTEM BUFFERS command. You can scroll through the list and specify a buffer you want to view by moving the cursor to the buffer name and pressing the Select key. EVE puts the buffer in your current window.

NOTE: Do not delete system buffers because these buffers are necessary for some commands to work properly.

8.1.7.2 Displaying the Contents of the Messages Buffer

EVE uses the Messages window, which appears at the bottom of the screen, to communicate error and informational messages during an editing session. The Messages window displays the last message in the Messages buffer.

You can display these messages with the BUFFER command. To display the contents of the Messages buffer, enter the command BUFFER MESSAGES. To return to the buffer you were editing, enter the BUFFER command followed by the name of the appropriate buffer.

8.1.7.3 Editing Two Buffers

During an editing session, you can use several buffers if you want to edit more than one file or if you want temporary storage areas for manipulating blocks of text. Multiple buffers are especially useful if you want to copy text from one file to another.

To create a new buffer, enter the GET FILE command and the name of the file you want to copy to the new buffer. You can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type. You can use the ellipsis wildcard ([. . .]) as a substitute for a directory specification.

If the specified file exists, EVE reads the contents of the file into a new buffer and displays the buffer in the current window. Otherwise, EVE creates an empty buffer and displays the buffer in the current window.

To change the buffer in the current window, press the DO key, type BUFFER and the name of the buffer you want to display on the screen, and press RETURN. If you forget a buffer name, enter the SHOW BUFFERS command to display the names of active buffers in your editing session, and specify a buffer with the Select key.

If you exit from an editing session in which you have modified multiple buffers, EVE writes the contents of the current buffer to a file and then asks you whether you want to write each of the other modified buffers to files.

8.1.7.4 Reading and Writing Files

There are three ways to read a file into an EVE buffer:

- Invoke EVE with a file specification.
- Enter the INCLUDE FILE command and the name of the file you want to include. EVE reads the entire contents of a file into a buffer just before the line where the cursor is located. Using the INCLUDE FILE command does not change the name of the buffer on the status line.
- Enter the GET FILE command and the name of the file you want to use. This command creates a new buffer and reads the contents of an existing file into the buffer. The name of the buffer on the status line is the same as the file name you specified with the GET FILE command.

To write the contents of the current buffer to a file, enter the WRITE FILE command. You can include a file specification with the WRITE FILE command. If you do not include a file specification, EVE writes the file using the input file specification. If you created the current buffer with the BUFFER command, EVE prompts you for a file specification to which it writes the file.

8.1.8 Using Windows

During an EVE editing session, the text buffer you are editing is displayed on the screen in a window. A highlighted status line appears at the bottom of a window identifying the name, current editing mode, and current direction of the buffer.

EVE allows you to view more than one window on your terminal screen at the same time. For example, you can have two windows in order to view and edit different sections of the same buffer.

The following table describes EVE commands used to create and manipulate windows:

Command	Effect in a Window Environment
SPLIT WINDOW	Splits the window that the cursor is in, forming two smaller windows. Adding an argument to the command allows you to divide the window into more than two parts. For example, SPLIT WINDOW 3 splits the window into 3 windows.
TWO WINDOWS	Synonymous with the SPLIT WINDOW command.
NEXT WINDOW	Puts the text cursor in the next (or other) window.
PREVIOUS WINDOW	Puts the text cursor in the previous (or other) window.
OTHER WINDOW	Synonymous with the NEXT WINDOW command.
ONE WINDOW	Restores the current window as a single, large window.
DELETE WINDOW	Deletes the window the text cursor is in, assuming you are using more than one window.
ENLARGE WINDOW	Enlarges the current window by a specified number of lines. For example, ENLARGE WINDOW 5 enlarges the window the text cursor is in by 5 lines. The adjacent window shrinks accordingly.
SHRINK WINDOW	Shrinks the current window by a specified number of lines. For example, SHRINK WINDOW 5 shrinks the window the text cursor is in by 5 lines. The adjacent window is enlarged accordingly.

8.1.8.1 Editing One Buffer

To view or edit two sections of a file at the same time, use the SPLIT WINDOW command. EVE splits your screen and creates two identical windows. The cursor maintains its position in the buffer but appears only in the bottom window. Notice that the buffer name in each of the status lines is the same.

Now you can edit different sections of a single file. Any edits that you make in one window are made simultaneously in the other window. Unless you are viewing two different sections of a file, you can see EVE incorporate edits simultaneously in the two windows.

Displaying two sections of a long file makes moving text within a file very efficient. You can select and remove text from one part of the file and insert it into the other. To move the cursor from one window to the other, enter the NEXT WINDOW command.

To remove the second window from the screen and expand the current window to occupy the whole editing area, press the DO key, type ONE WINDOW, and press RETURN.

8.1.8.2 Editing Two Buffers

The following steps describe how to edit two buffers containing different files:

1. Invoke EVE to edit a file.
2. Create two windows by entering the command SPLIT WINDOW. EVE splits your screen and creates two windows.
3. Put a different buffer in the current window using either the GET FILE or the BUFFER command.

   To create a new buffer in the current window, use the GET FILE command with a file specification.

   Or, to display a buffer that you created earlier in the editing session in the current window, enter the BUFFER command and the name of the buffer you want to display.

   EVE replaces the current buffer with the buffer named with the GET FILE command or the BUFFER command.
4. Your terminal screen now displays two different buffers. You can select and remove text from one buffer and insert it into the other buffer. To move the cursor from one window to the other, enter the command NEXT WINDOW.

The following example shows you how to edit two files and move text from one file to another using two windows. First, invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward

3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the DO key, type the command SPLIT WINDOW, and press RETURN to create two windows on your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.

 [End of file]


 Buffer RHYMES.DAT                                    | Insert | Forward

She rhymes with tree,
also with bee,
and this one makes three.

[End of file]


 Buffer RHYMES.DAT                                    | Insert | Forward
Command:  SPLIT WINDOW
```

Press the DO key, type the command GET FILE OLDFILE.DAT, and press RETURN to create a new buffer containing the text of OLDFILE.DAT in the bottom window of your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]


Buffer RHYMES.DAT                                     | Insert | Forward

 Schedule for 1 July
 10:00 AM meeting with supervisor
 Read and review memo from Donna
 Work on Pascal program
 [End of file]


Buffer OLDFILE.DAT                                    | Insert | Forward

 4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Move the cursor to the letter *R* in the word *Read,* press the Select key, and press the down arrow twice. The last two lines in OLDFILE.DAT are highlighted.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                                 | Insert | Forward

Move the text cursor to select text.
```

Press the Remove key to place the highlighted text in the Insert Here buffer.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer OLDFILE.DAT                                 | Insert | Forward

Remove completed.
```

Enter the NEXT WINDOW command to move the cursor to the other window. Move the cursor to the bottom of the buffer and press the Insert Here key. The text that you removed from OLDFILE.DAT is inserted into RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer OLDFILE.DAT                                   | Insert | Forward
```

Enter the ONE WINDOW command to remove all other windows from the screen and expand the window containing the cursor to occupy the whole editing area of the screen. Press the DO key, type ONE WINDOW, and press RETURN.

If you exit from the editing session, EVE writes the contents of the current buffer to a file. EVE prompts *Write buffer?* if another modified buffer exists. To write the contents of the other buffer to a file, type Y.

8.1.9 Defining Keys

You can define keys to execute EVE commands or to enter a series of keystrokes.

EVE does not allow you to define the RETURN key (CTRL/M), the space bar, or any printing characters (such as letters, digits, and punctuation marks) on the main keyboard. See the description of the DEFINE KEY command in the Reference Section for a list of other keys that you should not define.

8.1.9.1 Defining Keys to Execute an EVE Command

The DEFINE KEY command assigns an EVE command to a single key or control key sequence. You can, in effect, create your own editing keys to enter EVE commands that you use frequently. If you press a key to which you have assigned an EVE command, EVE provides the help text for that EVE command. Key definitions are discarded when you terminate an EVE editing session unless you use the SAVE EXTENDED EVE command (see Section 8.1.9.4) to save key definitions from one editing session to the next.

To define a key, do the following:

1. Press the DO key, enter the command DEFINE KEY, and press RETURN.
2. Type the EVE command that you want to assign to the key and press RETURN.
3. Press the key to be associated with the EVE command.

The message *Key defined* appears if you have successfully defined a key.

The following command assigns the FILL command to CTRL/F:

```
Command: DEFINE KEY=CTRL/F FILL
```

8.1.9.2 Defining Keys to Enter a Learn Sequence

The LEARN command assigns a sequence of keystrokes called a *learn sequence* to a single key or control key. Learn sequences allow you to enter the same series of keystrokes in a buffer any number of times by pressing one key. If you press a key to which you have assigned a learn sequence, EVE provides a HELP message stating that you have defined the key; the HELP diagram labels as *sequence* any keys to which you have assigned a learn sequence, but does not describe what the key sequence is. All learn sequences are discarded when you terminate an EVE editing session unless you use the SAVE EXTENDED EVE command (see Section 8.1.9.4).

Define a learn sequence as follows:

1. Press the DO key, enter the LEARN command, and press RETURN.
2. Type the keystrokes to be remembered.
3. Press CTRL/R followed by the key to be associated with the learn sequence.

The message *Key sequence remembered* appears if you have successfully defined a key.

The following example shows how to define a learn sequence that will insert a string of text into your file when you press CTRL/F. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

First, move to the end of the buffer. To begin the definition of the learn sequence, enter the LEARN command.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward
Command:  LEARN
```

Insert the text *And what is a rhyme?* at the end of your file. This is the text that EVE is to remember.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]

Buffer RHYMES.DAT                                    | Insert | Forward

Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

Press CTRL/R.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
 [End of file]

Buffer RHYMES.DAT                                    | Insert | Forward

Press the key that you want to use to see what was just learned:
```

Press CTRL/F, the key to which you are assigning the learn sequence.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
 [End of file]

Buffer RHYMES.DAT                                    | Insert | Forward

Key sequence remembered
```

For the rest of the editing session, press CTRL/F to insert the text *And what is a rhyme?* at the current cursor position.

8.1.9.3 Defining a GOLD Key

You can assign two editing functions to one editing key if you create a *GOLD* key. One editing function is performed by pressing the editing key. The other function is performed by first pressing the GOLD key and then pressing the same editing key. To define a GOLD key, enter the SET GOLD KEY command and press the key you want to use as the GOLD key. Once defined, the message *GOLD key set.* appears in the Messages buffer.

Once you have defined a GOLD key, you can use the GOLD editing keys that EVE predefines (on VT200-series and VT300-series terminals). To see a diagram of these commands, enter HELP KEYPAD. The GOLD editing keys appear in reverse video.

You can also create your own key definitions using the GOLD key. The following example demonstrates how to define a GOLD key and assign two commands to a single key. The example defines the number 4 key on the numeric keypad as the GOLD key and then assigns the BOTTOM and TOP commands to the CTRL/G key. Thus, pressing CTRL/G alone enters the BOTTOM command, and pressing the GOLD key followed by CTRL/G enters the TOP command.

Invoke EVE to edit the file RHYMES.DAT. Define a GOLD key by pressing DO, typing SET GOLD KEY, and pressing RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
 Buffer  RHYMES.DAT                                    | Insert | Forward
Command:   SET GOLD KEY
```

Press the number 4 key on the numeric keypad.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                     | Insert | Forward
Press the key that you want to use as the GOLD key:
GOLD key set
```

Press the DO key, type DEFINE KEY, and press RETURN.

```
 She rhymes with tree,
 also with bee,
 and this one makes three.
 [End of file]
 Buffer  RHYMES.DAT                                    | Insert | Forward
Command:   DEFINE KEY
```

Type BOTTOM and press RETURN.

```
And what is a rhyme?
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                                  | Insert | Forward
EVE command:  BOTTOM
```

Press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                                  | Insert | Forward

Key defined
```

Now define the GOLD CTRL/G key to enter the TOP command. Press the DO key, type DEFINE KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                                  | Insert | Forward
Command:  DEFINE KEY
```

Type TOP and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                                  | Insert | Forward
EVE Command:  TOP
```

Press the GOLD key (number 4 on the numeric keypad) and then press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                                  | Insert | Forward

Key defined
```

For the rest of your editing session, when you press CTRL/G, EVE executes the BOTTOM command; and when you press the GOLD key (number 4 on the numeric keypad) followed by CTRL/G, EVE executes the TOP command. If you press the GOLD key by mistake, press the Select key to cancel it.

To save key definitions from one editing session to the next, use the SAVE EXTENDED EVE command, described in Section 8.1.9.4.

8.1.9.4 Saving Key Definitions and Learn Sequences

The command definitions and learn sequences that you assign to keys extend the power of the EVE editor, making editing faster and more efficient. You can save your definitions cumulatively in a section file. EVE creates a section file automatically when you save definitions with the SAVE EXTENDED EVE command. Enter the SAVE EXTENDED EVE command before you terminate the editing session, using the following format:

SAVE EXTENDED EVE filename

The section file is saved in your current default directory unless you include a device and directory in the file specification. The file type defaults to TPU$SECTION. You can specify the same file specification each time you execute the SAVE EXTENDED EVE command. By doing this, you add any new key definitions and learn sequences to the same section file.

To use this extended version of EVE, you must include the /SECTION qualifier in the command line when you invoke EVE. For example, to invoke EVE to edit the file RHYMES.DAT using the section file WORKDISK:[USER]MYDEFS.TPU$SECTION, enter the following command:

```
$ EDIT/TPU/SECTION=WORKDISK:[USER]MYDEFS.TPU$SECTION RHYMES.DAT
```

Remember, you can define a command symbol to invoke this or any other lengthy command line.

8.1.10 Using DCL Within EVE

You can execute a DCL command from within EVE, or you can use a subprocess to switch between the DCL command level and an EVE editing session very quickly.

8.1.10.1 Executing a DCL Command

To execute a DCL command from within EVE, press the DO key, type the EVE command DCL and the DCL command you wish to execute, and press RETURN. The message *Creating DCL subprocess . . .* appears in the Message buffer.

8.1.10.2 Creating a Subprocess

You can create a subprocess to switch between an EVE editing session and DCL command level without terminating your editing session. To create a subprocess, press the DO key, type the SPAWN command, and press RETURN. EVE suspends the current editing session and connects your terminal to a new VMS subprocess. The DCL prompt ($) appears on your screen.

While the most common reasons to spawn a subprocess are to invoke the Mail Utility and to run screen-oriented programs, your subprocess can invoke any VMS utility or execute any DCL command.

To return to your editing session, log out of the subprocess by typing the DCL command LOGOUT and pressing RETURN. EVE resumes the editing session, and the cursor appears in the location it occupied before you spawned the subprocess.

8.2 The EDT Editor

EDT is an interactive text editor. With EDT you can create a new file, insert text into it, and modify that text. You can also edit text in existing files.

EDT provides both line and keypad editing. In line editing, you type the editing command and the range of text you want the command to affect. In keypad editing, you move the cursor directly to the text you want to change and press keypad keys to enter the editing commands.

8.2.1 Invoking and Terminating EDT

An editing session begins when you invoke EDT with the DCL command EDIT. In an editing session, you can create and edit a new file, or you can edit an existing file. The session ends when you enter the EXIT or QUIT command.

8.2.1.1 Invoking EDT

To invoke EDT, type the DCL command EDIT and specify as a parameter the file you want to edit. If the specified file already exists, EDT saves the existing versions and places a copy of the latest version in your buffer. (A buffer is the temporary storage area in which you edit text.) The existing versions of the file remain unchanged. For example, to edit an existing file named MEMO.TXT, enter the following command line:

```
$ EDIT MEMO.TXT

Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
```

The first few lines of the latest version of the file appear on the screen. The cursor is positioned at the top of the screen, and EDT is ready to receive a keypad-editing command.

If you invoke EDT to create a file, the following message appears:

```
$ EDIT NEWFILE.TXT

[EOB]


Input file does not exist
```

Only the EDT message and the end-of-buffer symbol, [EOB], appear on the screen, and EDT is ready to receive keypad-editing commands. See Section 8.2.2.1 for a description of EDT line commands.

NOTE: In the previous examples, you enter EDT in keypad (change) mode because a startup command file (SYS$LOGIN:EDTINI.EDT) containing the SET MODE CHANGE command has been executed. If this command is not executed in an

EDT startup command file, you will enter EDT in line mode. See Section 8.2.7.1 for more information on EDT startup command files.

8.2.1.2 Terminating EDT

To terminate an EDT session, press CTRL/Z. This puts you into line-editing mode. You can type EXIT or QUIT at the asterisk (*) prompt. QUIT terminates the editing session and does not save your edits. EXIT saves your edits in a new version of the file. (Note that the existing versions of a file remain unchanged regardless of how the editing session is terminated.)

To save your edited text, use the line-editing command EXIT to terminate EDT. When you enter the EXIT command, EDT creates an output file containing the edited version of the input file. By default, the output file has the same name and type as the input file, with the version number incremented by 1.

For example, if you enter the EXIT command after editing a file named MEMO.TXT;3, EDT creates a higher version named MEMO.TXT;4 as follows:

```
*EXIT
DISK1:[USER]MEMO.TXT;4  2 lines
$
```

To override the default output file name, enter the EXIT command with a new file specification as the parameter. For example, if you end the same editing session with EXIT MICE.TXT, EDT names the output file MICE.TXT;1, provided no other file named MICE.TXT exists.

```
*EXIT MICE.TXT
DISK1:[USER]MICE.TXT;1  2 lines
$
```

To terminate EDT without saving your edits, use the line-editing command QUIT. All edits you have made to the text are ignored, and no output file is created.

```
*QUIT
$
```

The QUIT command is a useful way to terminate EDT when you have opened a file by mistake. No new file version is created.

8.2.2 Entering EDT Commands

Enter most keypad-editing commands by pressing a keypad key. Enter line-editing commands by typing them after the line-editing prompt and pressing RETURN.

8.2.2.1 Entering EDT Line Commands

EDT prompts for line-editing commands with an asterisk. Line-editing commands usually operate on a range of one or more lines of text that you specify as a parameter for the command. For example, to display an entire file on your screen, enter the TYPE command and specify WHOLE as the parameter as follows:

```
*TYPE WHOLE
```

You can abbreviate EDT line-editing commands. For clarity, the examples in this chapter show complete line-editing commands.

8.2.2.2 Entering Keypad Commands

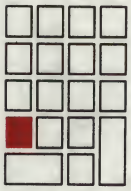
In keypad editing, the screen displays editing changes as you make them. You type text from the main keyboard and enter keypad-editing commands from the numeric keypad. (To initiate keypad editing, you must first enter the line-editing command CHANGE or have SET MODE CHANGE in your EDT startup file. See Section 8.2.4.2 for information on the CHANGE command.)

(See the description of EDT line-editing commands in the Reference Section for more information about keypad editing keys.)

Each key in the keypad performs at least one editing command; most perform two. Pressing a key invokes the regular, or upper, function. To invoke the alternate, or lower, function of a key, press the GOLD key (labeled PF1) first, followed by the desired key. In the examples that follow, a small diagram of the keypad highlights the key or keys that perform the command being described. The text associated with the keypad illustrates the effect of that editing command.

For example, keypad key 1 performs both the WORD and the CHNGCASE functions. To invoke the WORD command, press WORD: the cursor moves to the beginning of the next word.

WORD

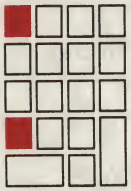


```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
```

To invoke the CHNGCASE command, press the GOLD key first and then CHNGCASE. The character at the cursor or the characters highlighted with the select key changes from lowercase to uppercase or from uppercase to lowercase.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
```

CHNGCASE



```
Once The weather turns cold, mice may find a crack in your foundation
and enter your house.  They're looking for food and shelter from the harsh
weather ahead.

[EOB]
```

The supplemental editing keys on the VT200 keypad perform the same functions as some of the EDT keypad keys. (See the description of EDT line-editing commands in the Reference Section for more information about these supplemental editing keys.)

8.2.2.3 Canceling EDT Commands

Use CTRL/C to cancel the currently executing EDT command without affecting previous edits. For example, to stop the display of a long file, press CTRL/C.

```
*TYPE WHOLE
    .
    .
    .
 CTRL/C
 CANCEL
Aborted by CTRL/C
*
```

The display stops and the CTRL/C message appears.

8.2.3 Getting HELP in EDT

EDT provides a help facility for each of the EDT editing modes.

8.2.3.1 Getting HELP with Keypad-Editing Commands

To display a diagram of the keypad keys and their functions, enter change mode (assuming you are in line-editing mode) and then press the HELP key (labeled PF2). (On VT200-series terminals, you can also use the HELP key on the supplemental editing keypad.) To display information about a particular keypad command, first press the HELP key and then press the keypad key.

8.2.3.2 Getting HELP with Line-Editing Commands

To display a list of EDT topics on which information is available, type HELP and press RETURN. To display information about a particular command or topic, type HELP followed by the name of the topic and press RETURN. EDT responds with a display of information about the topic and a list of related topics about which information is available. To display information about the use of a particular command qualifier, type HELP plus the command and that qualifier and press RETURN. For example, to display information on the use of /QUERY with the COPY command, enter the following command line:

```
*HELP COPY /QUERY
```

8.2.4 Changing Editing Modes

You can easily switch back and forth between line and keypad editing; you can also enter line-editing commands from keypad mode. Before using keypad commands, be sure that your terminal type is set properly. (Use SHOW TERMINAL to display the setting and SET TERMINAL/INQUIRE to set the terminal type.)

8.2.4.1 Changing from Keypad to Line Editing

To change from keypad editing to line editing, press CTRL/Z. The asterisk prompt appears at the bottom of your screen, indicating EDT is ready to accept line-editing commands.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
CTRL/Z


*
```

8.2.4.2 Changing from Line to Keypad Editing

To change from line editing to keypad editing, enter the CHANGE command:

```
*CHANGE
```

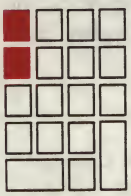
The first 22 lines of the file are displayed on your screen. If the file has fewer than 22 lines, the [EOB] symbol appears below the last line of the file.

8.2.4.3 Entering Line-Editing Commands from Keypad Mode

The keypad COMMAND function allows you to enter line-editing commands without leaving keypad mode. First, enter COMMAND (by pressing GOLD and then COMMAND) to invoke the *Command:* prompt, then type the line-editing command and press ENTER. (If you press RETURN by mistake, ^M appears; delete the ^M by pressing the DELETE key on the main keyboard, and press ENTER.) The following example enters the line-editing command SET QUIET, which suppresses the sound made when EDT issues an error message:

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
```

COMMAND



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.

[EOB]
Command:   SET QUIET
```

ENTER

8.2.5 Recovering from Interruptions

You can recover from interruptions to your editing session in the following ways:

- Deleting extraneous characters—Pressing CTRL/W removes extraneous characters (such as a broadcast message or a message indicating that you have received electronic mail) from the screen and restores the previous display. Use CTRL/W to ensure that the cursor is in the correct position.

- Resuming an interrupted editing session—The DCL command CONTINUE resumes an editing session that was interrupted by pressing CTRL/Y, so long as only built-in DCL commands were entered after pressing CTRL/Y. (See Chapter 1 for a list of built-in commands.) For example, you could press CTRL/Y, enter the command SHOW TIME, and return to your editing session with the CONTINUE command.

  (Press CTRL/W to refresh the screen display. The text of your editing session is once again displayed.)

- Recovering a lost session—By default, EDT keeps a journal file with the same file name as the input file and a file type of JOU. If the editing session ends without interruption, the journal file is deleted when you terminate the session. If the editing session is aborted (for example, during a system failure, in response to pressing CTRL/Y, or entering the QUIT/SAVE command), you can recover your edits (with the exception of those commands entered just prior to the interruption). Enter the same command line you used to begin the editing session, adding the /RECOVER qualifier. For example:

  ```
  $ EDIT/RECOVER MEMO.TXT
  ```

  EDT will reproduce the editing session, reading the commands from the journal file and executing them on the screen.

8.2.6 EDT Keypad Editing

While line editing allows you to manipulate large portions of text easily, keypad editing provides easy manipulation of small units of text. Several EDT keypad commands enable you to find, insert, delete, substitute, and move text in a file. The cursor can be moved through a file in a variety of ways, and the position of the cursor in a file determines how text will be affected by EDT commands.

8.2.6.1 Manipulating the Cursor

You can manipulate the cursor with commands that move it unit by unit through the text or with commands that move it directly to a particular location. Several commands that move the cursor are controlled by the ADVANCE and BACKUP commands, which set the cursor's direction forward and backward. Unless otherwise stated, this chapter assumes the default direction of the cursor to be ADVANCE.

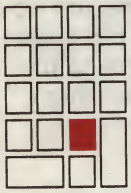
You can move the cursor by character, word, and line units.

Use one of the following keys to move the cursor by character:

- RIGHT ARROW — Moves the cursor one character to the right.
- LEFT ARROW — Moves the cursor one character to the left.
- CHAR — Moves the cursor one character in the current direction.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

CHAR

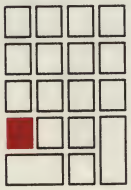


```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The WORD command moves the cursor to the beginning of the next or previous word.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

WORD



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The following keys move the cursor by line:

- UP ARROW—Moves the cursor up one line.
- DOWN ARROW—Moves the cursor down one line.
- EOL—Moves the cursor to the end of the current or previous line.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

EOL

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

- F12 (the BACKSPACE key on VT100-series terminals)—Moves the cursor to the beginning of the previous line.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.  [F12] ([BACKSPACE])

Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
```

- LINE—Moves the cursor to the beginning of the next line or previous line.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
```

LINE

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The OPEN LINE command terminates a line without moving the cursor. (The RETURN key also terminates a line, but moves the cursor to the next line.) The OPEN LINE command is useful when you want to insert a blank line or a new line of text. When the cursor is placed at the beginning of a line and the OPEN LINE command is entered, the text on that line is moved down so that the cursor is at the beginning of a blank line as follows:

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

OPEN LINE

```
Once the weather turns cold, mice may find a crack in your

foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

To move the cursor by large units, use the SECT and PAGE commands. The SECT and PAGE commands allow you to scan several lines of text at a time. The direction in which EDT moves depends upon whether ADVANCE or BACKUP is set.

- SECT—Moves the cursor across a 16-line section of text in EDT's current direction. If there are fewer than 16 lines, SECT moves the cursor across the existing lines.

  (On the VT200-series terminals, the supplemental editing keypad key Next Screen moves the cursor 16 lines forward, regardless of EDT's current direction. The supplemental editing keypad key Prev Screen moves the cursor 16 lines backward, regardless of EDT's current direction.)

- PAGE—Moves the cursor to the next or previous page boundary (form feed) or to the end or top of the buffer if there is no boundary. To insert form feeds in your text, use CTRL/L.

The TOP and BOTTOM commands allow you to move directly to the beginning or end of a buffer. (See Section 8.2.6.8 for more information about buffers.)

- TOP—Moves the cursor to the beginning, or top, of the buffer.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

TOP

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

- BOTTOM—Moves the cursor to the end, or bottom, of the buffer.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

BOTTOM

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

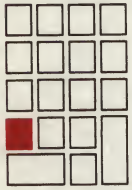
The ADVANCE and BACKUP commands control the cursor's direction for the following EDT keypad-editing commands: CHAR, CHNGCASE, EOL, FIND, FNDNXT, LINE, PAGE, SECT, SUBS, and WORD. Each of the directional commands remains in effect until you set the cursor in the opposite direction with the other command.

- ADVANCE—Sets the cursor's direction forward so that subsequent commands move the cursor in the forward direction. For example, if you enter the WORD command after using ADVANCE, the cursor moves forward one word.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

ADVANCE

WORD



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

- BACKUP—Sets the cursor's direction in the backward direction so that subsequent commands move the cursor toward the top of the buffer. For example, if you enter the WORD command after using BACKUP, the cursor moves backward one word.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

BACKUP

WORD

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The cursor remains set in the backward direction until you press ADVANCE. For example, if you enter a second WORD command in the preceding example you receive a message indicating that the command requests EDT to back up past the top of the buffer.

The ADVANCE and BACKUP commands are particularly important in string searches; see Section 8.2.6.4 for more information on searches.

8.2.6.2 Inserting Text

To insert text in EDT keypad editing, position the cursor where you want the text to be inserted and begin typing; the cursor remains one position to the right of the last character inserted. Inserting text in the middle of a line moves both the cursor and the remainder of the line one position to the right for each character inserted. When the line exceeds 80 characters, the text you type will either wrap to the following line or disappear off your screen, depending on the status of the SET SCREEN, SET [NO]TRUNCATE, and SET [NO]WRAP commands. (See Section 8.2.7.2 for information about screen formatting commands.)

8.2.6.3 Deleting and Restoring Text

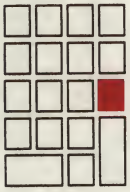
The delete commands work like the cursor movement commands. In EDT keypad editing, you can delete by character using the Delete key (<X]) (DELETE on VT100-series terminals) and DEL C; by word using F13 (LINEFEED on VT100-series terminals) and DEL W; and by line using DEL L, DEL EOL, and CTRL/U.

The deleted text is stored in a buffer so that you can also restore the character (UND C), word (UND W), or line (UND L) most recently deleted wherever and as many times as you need. Note that the undelete commands restore only the corresponding units of text that were most recently deleted. For example, if you have deleted two lines of text with the DEL L (delete line) command, the UND L (undelete line) command will restore only one line, the line most recently deleted.

The <X] key on the main keyboard (the DELETE key on VT100-series terminals) deletes the character immediately to the left of the cursor. The EDT keypad-editing command DEL C deletes the character directly at the cursor. The UND C command restores the last character deleted with either the <X] (DELETE) key or the DEL C command. For example:

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

DEL C



```
nce the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

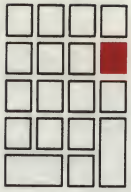
UND C

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The F13 key on the main keyboard (the LINEFEED key on VT100-series terminals) deletes to the beginning of the current or preceding word. The DEL W command deletes to the end of the current word. Blank spaces are considered part of the word they follow, while all other word delimiters are considered to be separate words. The UND W command restores the last word deleted with either the F13 (LINEFEED) key or the DEL W command. For example:

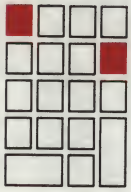
```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

DEL W



```
Once the weather turns cold, may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

UND W



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

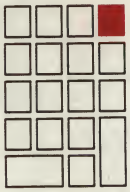
The following commands delete a line (or part of a line) of text:

- DEL L—Deletes from the cursor to the end of the line, including the line terminator. If the cursor is at the beginning of the line, the entire line is deleted, and the cursor is positioned at the beginning of the next line.
- DEL EOL—Deletes from the cursor to the end of the line (excluding the line terminator), leaving the cursor at the end of the truncated line.
- CTRL/U—Deletes from the cursor to the next previous beginning of line, leaving the cursor at the beginning of the previous line. (If CTRL/U is used when the cursor is at the beginning of the line, the previous line is deleted.)

The UND L command restores the last line (or part of a line) that was deleted with the DEL L, DEL EOL, or CTRL/U command. For example:

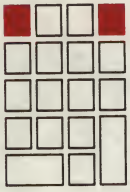
```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

DEL L



```
Once the weather foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

UND L



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.
[EOB]
```

The EDT line-editing command DELETE is useful for deleting large sections of text. Generally, you use line numbers to specify a range for a line-editing command. For example, to delete lines 306 through 860, enter the following:

```
*DELETE 306 THRU 860
555 lines deleted
    861    Rodents have had a profound effect on human civilization.
*
```

Note that the EDT line-editing command SET NUMBERS (the default) must be in effect for line numbers to be displayed in EDT line editing.

You can also use certain keywords (such as WHOLE, REST, BEFORE) as range specifiers. For example, if you are in the middle of a long buffer and want to delete from the cursor to the end of the buffer, enter the following:

```
*DELETE REST
43 lines deleted
[EOB]
*
```

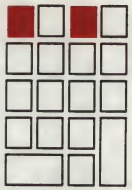
(You can also specify range by using the EDT keypad-editing command SELECT. See Section 8.2.6.7 for information on SELECT.)

8.2.6.4 Locating Text

You can move the cursor to a character string you specify with the FIND and FNDNXT EDT keypad-editing commands. The FIND command searches for the specified character string between the current position of the cursor and the beginning or end of the buffer (depending on whether the ADVANCE or the BACKUP command is in control). EDT does not distinguish between uppercase and lowercase letters unless you use the SET SEARCH EXACT line-editing command. When EDT finds the string, it positions the cursor at the first character in the string (unless the SET SEARCH END command is in effect, and the cursor is positioned at the last character in the string). In a long file, the message "Working" may flash on the screen while EDT searches for the string.

For example, to delete a comma after the word "house" in the following text, you can use the FIND command to move the cursor to the string "house." First, enter the EDT keypad command FIND by pressing the GOLD key and then the FIND key (on the VT200-series terminal you can also use the FIND key located on the supplemental editing keypad). Next, type the string you want to locate (the search string) after the Search for: prompt.

FIND



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.  Once inside, they may gnaw
through electrical wires and raid your food. Because mice reproduce
so quickly, what started as one or two mice can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
Search for:   house
```

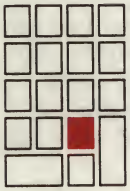
To search in the forward direction, use the ADVANCE command to enter the search string.

ADVANCE

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  Because mice reproduce so
quickly, what started as one or two mice can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
```

Use the CHAR command to move the cursor to the comma after the word "house." Then use the DEL C command to delete the comma.

CHAR



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  Because mice reproduce so
quickly, what started as one or two mice can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house you can prevent these rodents from ever getting in.
[EOB]
```

To find the next occurrence of the string located with the FIND command, use the FNDNXT (find next) command. If there is no other occurrence of the string (as in the example above), EDT issues the message "String was not found."

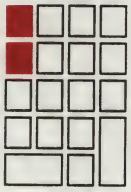
NOTE: Note that the directional setting of the cursor determines the direction of the search. After you press FIND, you can press either ADVANCE or BACKUP (depending on the direction in which you want to search) to enter the search string. You can also use the ENTER command, which applies the current direction to the search.

8.2.6.5 Substituting Text

To substitute one character string for another, you can use the SUBS keypad-editing command or the SUBSTITUTE line-editing command. The EDT line-editing command can make global substitutions; that is, it can replace every occurrence of one character string in the specified range with another string using only one EDT line-editing command. In contrast, you must use the keypad SUBS command (press the GOLD key followed by the SUBS key) for each substitution you make. (If you do not specify a range, the line-editing command SUBSTITUTE replaces only the first occurrence of the search string in the current line with the substitute string.)

For example, to substitute the string "mice" for "elephants" throughout a buffer, enter the line-editing command SUBSTITUTE, the old string, and the new string, separating all three with the same delimiter. You can use any nonalphanumeric character (except the percent sign and underscore) as a delimiter for the SUBSTITUTE command, as long as the delimiting character is not part of either string. To apply the command to the entire buffer in a global substitution, specify WHOLE as the parameter. When the operation has been completed, EDT displays each occurrence of the substitution and the total number of substitutions. The following example substitutes the string "mice" for each occurrence of the string "elephants" in the following text:

COMMAND



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  Because elephants reproduce so
quickly, what started as one or two elephants can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
```

ENTER

```
1   Once the weather turns cold, elephants may find a crack
4   in your electrical wires and raid your food.  Because elephants reproduce
5   so quickly, what started as one or two elephants can quickly become an
3 substitutions
Press return to continue

Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  Because elephants reproduce so
quickly, what started as one or two elephants can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
```

Note that a global substitution replaces all occurrences of the string, regardless of case or surrounding characters. If you want EDT to search for exact comparisons of case, use the SET SEARCH EXACT command. If the search string occurs in the middle of a longer string, the substitution will still be made. For instance, a global substitution of "IN" for "AT" would change all words containing the string "AT." ("LATER" would become "LINER", "THAT" would become "THIN", "SAT" would become "SIN", and so on.)

To get EDT to prompt you before each substitution, use the /QUERY qualifier with the SUBSTITUTE command.

```
Command:   SUBSTITUTE\AT\IN\WHOLE/QUERY
```

EDT prompts you with a question mark (?) to verify each substitution. You can respond with one of the following:

Y Yes, do the substitution.

N No, do not do the substitution.

Q Quit, terminate the command.

A All, do the rest of the substitutions without query.

8.2.6.6 Moving Text

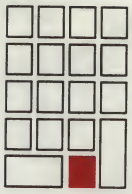
Both EDT keypad and line commands can move text; however, only line-editing commands transfer text between buffers and files.

8.2.6.7 Moving Text Within the File

The EDT keypad-editing command CUT deletes a selected range of text and the PASTE command inserts it at the cursor's current position. (On the VT200-series terminals, the supplemental editing keys Remove and Insert Here perform the same functions as the EDT keypad commands CUT and PASTE.) For instance, to move the first sentence in the second paragraph of the example to the end of that paragraph, move the cursor to the beginning of the sentence and press SELECT. (On the VT200-series terminals, the supplemental editing key SELECT performs the same function as the EDT keypad command SELECT.) This marks the beginning of the selected range. (You can cancel the SELECT command with the RESET command.)

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house.  They're looking for food and
shelter from the harsh weather ahead.  Once inside, they may gnaw
through electrical wires and raid your food. Because mice reproduce
so quickly, what started as one or two mice can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
```

SELECT

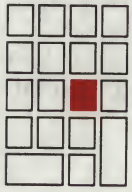


To mark the end of the selected range, move the cursor to the end of the sentence. The terminal highlights a selected range in reverse video. (The selected range includes the text up to the character preceding the cursor.)

```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  Because elephants reproduce so
quickly, what started as one or two elephants can quickly become an
invasion.  If you seal the cracks and holes on the exterior of your
house, you can prevent these rodents from ever getting in.
[EOB]
```

Press CUT to delete the selected text.

CUT

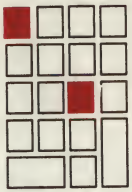


```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in.
[EOB]
```

Deleted text remains in the PASTE buffer until you perform another CUT operation. To restore the text, move the cursor to the appropriate position and enter the PASTE command. (The text will be inserted directly in front of the cursor.)

```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in.
[EOB]
```

PASTE

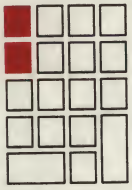


```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.
[EOB]
```

Because the selected text is held in the PASTE buffer until you perform another CUT operation (or give the line-editing command CLEAR PASTE), you can paste the text contained in the PASTE buffer as many times as you want. You can also enter the PASTE buffer to edit the text it contains. (See Section 8.2.6.8 for information on using multiple buffers.)

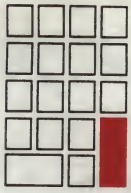
After moving the text, you may want to use the FILL command to reorganize selected text so that the maximum number of whole words are fitted within the current line width. The default line width is 80 characters, but you can use the SET WRAP command to use another line length for filling text. For example, you can set the line length to 71 characters with the EDT line-editing command SET WRAP and then fill a selected range of text.

COMMAND



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.
[EOB]
Command:   SET WRAP 71
```

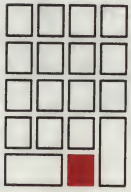
ENTER



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.
[EOB]
```

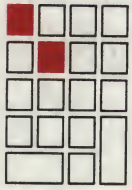
EDT will now wrap lines of inserted text and fill lines of selected text at a line width of 71 characters. Use the SELECT command to mark the text you want to affect and then enter the EDT keypad command FILL.

SELECT



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as
one or two elephants can quickly become an invasion.
[EOB]
```

FILL



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.   If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.
[EOB]
```

There are several EDT line-editing commands that move text. For example, the MOVE and COPY commands each perform a function similar to those of the keypad CUT and PASTE operations. MOVE deletes text from one location and inserts it in another; COPY inserts a copy of the text where specified without deleting any text. The EDT line-editing commands INCLUDE and WRITE perform tasks not possible with EDT keypad-editing commands:

- INCLUDE—Copies a file into the buffer you are currently editing or the buffer you specify. Follow the VMS conventions for file specifications when specifying the file to be copied to the buffer. For example, the following command copies the file named MEM.DAT to the buffer named BUF1:

  ```
  Command:   INCLUDE MEM.DAT =BUF1
  ```

- WRITE—Copies a specified range of text from a buffer (the current buffer by default) to a specified file. If you do not specify a range, the WRITE command copies the entire contents of the current buffer. For example, the following command copies the contents of the current buffer to the file ANIMALS.TXT:

  ```
  Command:   WRITE ANIMALS.TXT
  $DISK1:[USER]ANIMALS.TXT;1  11 lines
  ```

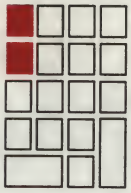
  The message displays the new file specification and length.

8.2.6.8 Using Multiple Buffers

When you begin editing a file with EDT, you are working on a copy of the file in a buffer called MAIN. (EDT also uses a buffer called PASTE to store the text that you delete with the CUT and APPEND commands; you can edit this buffer just as you can edit other text buffers.) You can create other buffers to store pieces of text during your EDT editing session. You can enter and edit these buffers; you can copy text to and from them; and you can write their contents to specified files.

To create a buffer, press the COMMAND key. Type the line-editing command FIND followed by the equal sign and the name you are giving the buffer, then press the ENTER key. For example, the following command creates a buffer named BUF1:

COMMAND



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.
[EOB]
Command:   FIND=BUF1
```

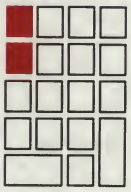
When you enter this command, the system responds by displaying only the [EOB] symbol, which indicates that the current buffer, BUF1, is empty. You can now insert and edit text just as you would in the MAIN buffer. To return to the MAIN buffer, follow the same procedure, typing FIND=MAIN rather than FIND=BUF1. To return to your previous position in the MAIN buffer, include a period after the buffer's name as follows:

```
Command:   FIND=MAIN.
```

The buffer named BUF1 remains intact until you exit from EDT, regardless of whether you enter the EXIT or QUIT command. That is, you can enter, edit, and exit from a buffer as necessary. However, when you exit from EDT, only the buffer MAIN is saved.

The SHOW BUFFER command displays the number of lines contained in each buffer and indicates (with an equal sign) the current buffer. The following example indicates that there are three buffers (including MAIN and PASTE, which always exist) and that MAIN is the current buffer:

COMMAND

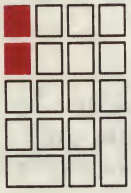


```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.


[EOB]
Command:   SHOW BUFFER
=MAIN  11   lines
 PASTE  3   lines
 BUF1   2   lines
Press return to continue
```

Pressing the RETURN key returns the cursor to its previous position in the buffer.

You can further manipulate the contents of a buffer by specifying the buffer's name in an EDT line-editing command. For example, if you are in the MAIN buffer and want to save the contents of BUF1 in a file named RODENT.TXT before exiting from EDT, enter the following command:

COMMAND



```
Once the weather turns cold, elephants may find a crack in your
foundation and enter your house.  They're looking for food and shelter
from the harsh weather ahead.  Once inside, they may gnaw through
electrical wires and raid your food.  If you seal the cracks and holes
on the exterior of your house, you can prevent these rodents from ever
getting in. Because elephants reproduce so quickly, what started as one
or two elephants can quickly become an invasion.


[EOB]
Command:   WRITE RODENT.TXT =BUF1

$DISK1:[USER]RODENT.TXT;1  2 lines
```

EDT returns a message indicating that the file has been created, and the cursor is returned to its previous location in the buffer.

8.2.7 Controlling EDT Sessions

You can control some of the characteristics of an EDT editing session with the SET commands. You can also define a macro (a sequence of line-editing commands) and define keys in EDT. You can enter these control commands interactively, or you can include them in an EDT startup command file.

8.2.7.1 Startup Command Files

An EDT startup command file contains EDT line-editing commands that are executed when you invoke EDT before you receive control of the editor. A startup command file is useful for setup operations in EDT; it can include specifications for screen format, definitions of text entities, and definitions of keys and macros. Generally, EDT reads a systemwide startup command file at the beginning of your editing session. If no systemwide startup command file exists on your system, EDT looks for a file named EDTINI.EDT in your default directory and processes the commands in that file.

To create an EDTINI.EDT file, invoke an editor and specify EDTINI.EDT as the file specification as follows:

```
$ EDIT EDTINI.EDT
```

Now list the commands, one per line. Some typical commands you might want to put in a startup command file follow:

```
SET QUIET
SET WRAP 60
SET SEARCH BOUNDED
SET TAB 5
DEFINE KEY GOLD P AS "PAR."
SET MODE CHANGE
```

When you exit from the editor, EDTINI.EDT is saved in your default directory. Every time you invoke the editor, the commands in your EDTINI.EDT file are in effect.

To specify an EDT startup command file named something other than EDTINI.EDT, you must include the file specification in the EDIT command line n as follows:

```
$ EDIT/COMMAND=startup-file-spec  file-spec
```

8.2.7.2 Controlling Screen Format with SET Commands

Several EDT commands control the format of a screen display. Some are listed below. See the EDT commands in the Reference Section for a comprehensive list of the SET commands.

- SET LINES n—Controls the number of lines that EDT displays on the screen. This number, which can be set from 1 to 22, defaults to 22. To set the screen to 15 lines, for example, type:

  ```
  Command:  SET LINES 15
  ```

  Note that if you are editing at slow baud rates, setting the number of lines low will increase your editing speed.

- SET SCREEN width—Controls the maximum length of the line EDT displays; the default width is 80 characters. (When there are more characters than the SET SCREEN command specifies, EDT displays a diamond at the end of the line.)

  ```
  Command:  SET SCREEN 132
  ```

  If you use the SET SCREEN command to make the screen wider than 80 on either a VT100- or VT200-series terminal, EDT changes the terminal's screen width to 132.

- SET [NO]TRUNCATE—Controls whether the characters that exceed the SET SCREEN width are displayed on the next line. The default is SET TRUNCATE, which ends the display of a line at the value of SET SCREEN.

  ```
  Command:  SET [NO]TRUNCATE
  ```

- SET [NO]WRAP n—Specifies n character positions as the point at which text will be moved to the beginning of the next line. When you are inserting text in EDT keypad mode and the cursor position exceeds the value of n, EDT wraps the next full word to the next line. (However, when you insert text in the middle of a line, that line does not always wrap.) The default is NOWRAP. To wrap the text exceeding 75 characters, for example, type:

  ```
  Command:  SET WRAP 75
  ```

The SET commands have corresponding SHOW commands; see the EDT commands in the Reference Section for a list of SHOW commands.

8.2.7.3 Controlling Editing Functions with SET Commands

Several commands control EDT's responses during an editing session, as follows. (See the EDT commands in the Reference Section for a comprehensive list of the SET commands.)

- SET ENTITY—Defines boundaries for the WORD, SENTENCE, PARAGRAPH, and PAGE entities. (The SENTENCE and PARAGRAPH entities are not used by any default key definitions; consequently, they are useful only in the key definitions you create with the DEFINE KEY command.) For example, the default boundaries for the WORD entity are a line feed, tab, form feed, line terminator, and space. To make the period and comma the only delimiters of the word entity, enter the following SET ENTITY command:

```
Command:   SET ENTITY WORD ',.'
```

- SET MODE—Controls the EDT editing mode to be entered when the processing of the EDTINI.EDT file is completed (either line or change mode, which is keypad mode). For example, to enter change mode instead of line mode at the beginning of editing sessions, insert the following command at the end of your EDT startup command file:

```
SET MODE CHANGE
```

- SET QUIET—Suppresses the sound made when EDT issues an error message in keypad mode. The default is NOQUIET.

8.2.7.4 Defining EDT Macros

An EDT macro allows you to execute a sequence of EDT line-editing commands whenever you invoke the macro. To define a macro, use the EDT line-editing command DEFINE MACRO to define the name of a buffer as the macro name. Then create and enter a buffer with the same name as the macro. (See Section 8.2.6.8 for information about using multiple buffers.) Once in the buffer, type the EDT line-editing commands in the desired sequence, one command per line. For example, the following macro inserts a four-line heading:

```
INSERT;NAME:
INSERT;DEPT:
INSERT;DATE:
INSERT;SUBJ:
[EOB]
```

Then exit from the buffer. To invoke the macro, enter its name as an EDT line-editing command. The lines of the heading are inserted at the cursor position:

```
NAME:
DEPT:
DATE:
SUBJ:
```

To make a macro available during other editing sessions, you can place the DEFINE MACRO command and the macro command sequence in an EDT startup command file. When you include a macro definition in a startup command file, be sure the command sequence contains the commands for entering the macro buffer (FIND=buffer-name.) and returning to the MAIN buffer (FIND=MAIN.). Note that you must precede each command in the sequence with the INSERT command. For more information about macro definitions, see Section 8.2.7.4.

Chapter 9

Processing Files with DIGITAL Standard Runoff

DIGITAL Standard Runoff (DSR) is a text formatter that processes source files into formatted text and intermediate files, and creates tables of contents and indexes. You process the source and intermediate files with the DCL commands RUNOFF, RUNOFF/CONTENTS, and RUNOFF/INDEX. With DSR, you can produce several types of documents including a memo, letter, and full-length manuscript.

The DSR source file is a file you created with EDT, EVE, or another text editor. This file has a default file type of RNO and contains text, DSR formatting commands, flags, and control characters. (DSR flags are special characters that you insert in text to specify, for example, emphasis of text, case of characters, and spacing of characters.) When the source file is processed, the DSR commands cause the text to be formatted into sections, paragraphs, lists, and so on. You can direct DSR to provide title pages, footnotes, tables of contents, indexes, and appendixes for the source files it processes by inserting control characters and other special identifiers within your text.

See the DSR commands in the Reference Section for rules on entering DSR commands and a description of each command.

DSR commands start with the control flag, which by default is represented by a period. DSR commands can be typed in uppercase, lowercase, or a combination of uppercase and lowercase. You can abbreviate DSR command names, but the abbreviations must be exactly as listed in the description of the DSR commands in the Reference Section.

9.1 Formatting Text

Using DSR, you can format text into paragraphs or lists, or maintain a literal display. You can adjust the margins and the spacing between lines or blocks of lines. DSR also provides commands for leaving blocks of space on a page and for writing notes and footnotes.

You can control such features as bolding and underlining with embedded flags. By default, DSR treats certain characters as flags rather than text. For example, by default, a character is underlined if it is preceded by an ampersand (&). (To place an actual ampersand in the text, you must enter an ampersand preceded by an underscore (_&) flag, or you must turn off the flag governing that character.) One approach is to turn off all flags at the start of your DSR file. The following commands turn off all flags that are on by default:

Flag Command	Flag Character
.NO FLAGS ACCEPT	Underscore (_) by default
.NO FLAGS COMMENT	Exclamation point (!) by default
.NO FLAGS LOWERCASE	Backslash (\) by default
.NO FLAGS SPACE	Number sign (#) by default
.NO FLAGS SUBINDEX	Right angle bracket (>) by default
.NO FLAGS UNDERLINE	Ampersand (&) by default
.NO FLAGS UPPERCASE	Circumflex (^) by default

When you need a particular flag, set the flag on (.FLAGS command), write the text that uses the flag, and set the flag off (.NO FLAGS command). (See Section 9.1.9.) You can change the special flag character when you specify the .FLAGS command.

Following are several examples that illustrate how to use various DSR commands to produce formatted output:

```
.BREAK
.BLANK
.BLANK 2
.RIGHT MARGIN 34
.CENTER;Twelve Days of Dieting
```

In the previous example, the .BREAK command ends the current line. The .BLANK command without a parameter inserts a blank line in the text. The .BLANK command with the parameter 2 inserts two blank lines in the text. The .RIGHT MARGIN command sets the right margin at position 34. The .CENTER command centers the text following the semicolon.

On any line containing a DSR command, the first item on the line must be a DSR command and the control flag must occupy position 1. Depending on the particular commands, a line containing a command may contain additional commands and text. In the following example, the DSR command occupies its own line. The end of the line terminates the command.

```
.BLANK 2
```

In the following example, the command and text are placed on one line. The semicolon acts as the command terminator.

```
.CENTER;Twelve Days of Dieting
```

In the following example, two commands are placed on one line. The beginning of the second command terminates the first command.

```
.BLANK 2.CENTER;Twelve Days of Dieting
```

The following example demonstrates the use of the DSR commands .BLANK and .CENTER to format text:

```
.RIGHT MARGIN 34
.CENTER;Twelve Days of Dieting
.CENTER;Watching Your Weight Increase
.BLANK 2
On the twelfth day of dieting, Millitsa gave to me,
.BREAK
Twelve hot fudge sundaes,
.BREAK
Eleven Hostess Twinkies,
.BREAK
Ten cherry cheese cakes,
.BLANK
Nine ladyfingers,
.BREAK
Eight date nut muffins,
.BREAK
Seven oatmeal cookies,
.BREAK
Six bags of Fritos,
.BREAK
Five coffee rings,
.BLANK
Four sticky buns,
.BREAK
Three Clark bars,
.BREAK
Two marbled cakes,
.BREAK
And a pizza with pepperoni.
```

The preceding lines of text coded in DSR produce the following output:

```
        Twelve Days of Dieting
      Watching Your Weight Increase

On the twelfth day of dieting, Millitsa gave to me,
Twelve hot fudge sundaes,
Eleven Hostess Twinkies,
Ten cherry cheese cakes,

Nine ladyfingers,
Eight date nut muffins,
Seven oatmeal cookies,
Six bags of Fritos,
Five coffee rings,

Four sticky buns,
Three Clark bars,
Two marbled cakes,
And a pizza with pepperoni.
```

Do not edit the output file from a DSR operation. Instead, modify the DSR file and reprocess it. If you do make minor modifications to the output file, the file does not have the carriage return record attribute (which causes each record in the file to produce a new line automatically when the file is displayed or printed). The carriage return and line feed control characters are embedded in the file.

9.1.1 Filling and Justifying Text

By default, DSR fills and justifies text. Filling adds words to each output line until the addition of another word would exceed the right margin. Justification inserts additional space between words on a line so that the last word reaches the right margin. The following example demonstrates how text is filled and justified:

```
.RIGHT MARGIN 45
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

The preceding lines of text coded in DSR produce the following output:

```
For it so falls out,  That  what we have
we prize not to the worth while we enjoy
it; but being lacked, lacked  and  lost,
Why, then we rack the value.
```

If you do not want filling or justification, you must explicitly specify the .NO FILL command or the .NO JUSTIFY command before you write the text. Turn filling or justifying back on by specifying the .FILL or .JUSTIFY command. The following example turns off justification but retains filling, which produces a ragged right margin:

```
.RIGHT MARGIN 45
.NO JUSTIFY
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
.JUSTIFY
```

The preceding lines of text coded in DSR produce the following output:

```
For it so falls out, That what we have
we prize not to the worth while we enjoy
it; but being lacked, lacked and lost,
Why, then we rack the value.
```

The next example turns off both filling and justifying, which formats the output lines in the same way as the input lines:

```
.NO FILL.NO JUSTIFY
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
.FILL.JUSTIFY
```

The preceding lines of text coded in DSR produce the following output:

```
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

9.1.2 Adjusting Margins and Centering Text

By default, margins are set at 0 and 70. Change the margin settings with the .LEFT MARGIN and .RIGHT MARGIN commands. The following example changes the right margin to position 60:

```
.RIGHT MARGIN 60
    .
    .
    .
```

To indent one or more lines of text on the left, increase the left margin setting. After you write the indented text, decrease the left margin setting by the same amount. Indent on the right by decreasing the right margin setting, writing the text, and increasing the right margin setting. The following example indents text by 10 spaces on either margin:

```
    .
    .
    .
.LEFT MARGIN +10.RIGHT MARGIN -10
indented text
.LEFT MARGIN -10.RIGHT MARGIN +10
    .
    .
    .
```

You can indent a single line of text from the left margin with the .INDENT command. The following example indents a line of text eight spaces:

```
.INDENT 8
I wandered lonely as a cloud
```

Indent a single line of text from the right margin with the .RIGHT command. You can also use the .RIGHT command to position a single line of text against the right margin as follows:

```
.RIGHT 0
I wandered lonely as a cloud
```

To center text between two margins, use the .CENTER command. Place the text to be centered on the line following the command as follows:

```
.CENTER
I wandered lonely as a cloud
```

You can also end the .CENTER command with a semicolon, and place the text to be centered on the same line as follows:

```
.CENTER;I wandered lonely as a cloud
```

9.1.3 Formatting Paragraphs

To separate text into paragraphs, place the .PARAGRAPH command between paragraphs. By default, the .PARAGRAPH command indents the first line of a paragraph five spaces, inserts one blank line before starting a new paragraph, and tests to ensure that room remains on the page for at least four lines of text. You can change the parameter values when you type your first .PARAGRAPH command or by placing a .SET PARAGRAPH command at the top of the file. The following example does not indent the first line of each paragraph and ensures that room remains on the page for at least three lines of text:

```
.SET PARAGRAPH 0,1,1
paragraph of text
.PARAGRAPH
paragraph of text
.PARAGRAPH
    .
    .
    .
```

You can also separate text by inserting .BLANK or .SKIP commands. The .BLANK command inserts the exact number of lines specified by the parameter (which defaults to 1). The .BLANK command does not provide for indentation or for testing the room left on the page; perform these actions with the .INDENT and .TEST PAGE commands. The following example separates two blocks of text with one blank line after first testing to ensure that at least three lines remain on the page:

```
paragraph of text
.TEST PAGE 3
.BLANK
paragraph of text
```

The .SKIP command takes into account the spacing you have in effect. When the default is in effect (spacing is 1), .BLANK and .SKIP are equivalent. However, if multiple spacing is in effect, the .SKIP command multiplies the skip value by the spacing value. You can specify something other than single spacing with the .SPACING command. (The .SPACING command also affects the test page value in the .PARAGRAPH and .SET PARAGRAPH commands.) The following example demonstrates double spacing, with two extra lines between blocks of text:

```
.SPACING 2
block of text
.SKIP
block of text
.SKIP
    .
    .
    .
```

9.1.4 Formatting Literal Text

To have text appear in the output file exactly as it appears in the DSR source file, enclose it with .LITERAL and .END LITERAL commands as follows:

```
.RIGHT MARGIN 34
.BLANK 2
.LITERAL
          Twelve Days of Dieting
       Watching Your Weight Increase

On the twelfth day of dieting, Millitsa gave to me,
Twelve hot fudge sundaes,
Eleven Hostess Twinkies,
Ten cherry cheese cakes,

Nine lady fingers,
Eight date nut muffins,
Seven oatmeal cookies,
Six bags of Fritos,
Five coffee rings,

Four sticky buns,
Three Clark bars,
Two marbled cakes,
And a pizza with pepperoni.
.END LITERAL
```

The preceding lines coded in DSR produce the following output:

```
          Twelve Days of Dieting
       Watching Your Weight Increase

On the twelfth day of dieting, Millitsa gave to me,
Twelve hot fudge sundaes,
Eleven Hostess Twinkies,
Ten cherry cheese cakes,

Nine lady fingers,
Eight date nut muffins,
Seven oatmeal cookies,
Six bags of Fritos,
Five coffee rings,

Four sticky buns,
Three Clark bars,
Two marbled cakes,
And a pizza with pepperoni.
```

Literal text is not filled and not justified; lines are the same as in the input file. Except for the .END LITERAL command, commands and flags are not recognized in literal text. In addition, commands and flags placed before the literal text do not affect the literal text except that you can set the left margin, set tab stops, set spacing, start bolding, and start underlining. You cannot reset any of these items until the literal text ends (for example, you would have to bold the entire literal block). The following example indents the literal text (the .MARGIN commands must be outside the literal block):

```
the text filled and justified
.LEFT MARGIN +5
.LITERAL
the literal text
.END LITERAL
.LEFT MARGIN -5
```

If you want the positional effect of literal text but also want to use commands and flags, you can turn off filling and justifying (.NO FILL and .NO JUSTIFY) and turn it back on after the literal text. If the literal text contains blank lines, specify .KEEP when you turn off the filling and justification as follows:

```
the text filled and justified
.NO FILL
.NO JUSTIFY
.KEEP
the literal text
.FILL
.JUSTIFY
.NO KEEP
```

For a small number of lines, you might place .BREAK or .BLANK commands (specify .BLANK 0 for single spacing) between the literal lines.

9.1.5 Formatting Lists

You can format text as lists. Although by default DSR creates a numbered list, you can create a bulleted or lettered list or use any other character or symbol to precede the list elements.

9.1.5.1 Numbered Lists

The following three commands format a numbered list:

- .LIST—Starts the list by leaving one blank line and indenting. The text of the list is indented nine spaces if the left margin is currently 0. Otherwise, the text of the list is indented four spaces.
- .LIST ELEMENT—Starts an element (an item) within the list. You can have any number of elements.
- .END LIST—Ends the list and writes a blank line.

Each element is preceded by a number starting with 1. Elements are separated by blank lines as follows:

```
.LIST
.LIST ELEMENT;grosbeak
.LIST ELEMENT;goldfinch
.LIST ELEMENT;redpoll
.LIST ELEMENT;sparrow
.END LIST
```

The preceding lines of text coded in DSR produce the following output:

```
1. grosbeak

2. goldfinch

3. redpoll

4. sparrow
```

Text for each list element can be placed after the semicolon terminating the command (as shown in the example) or can be placed on a line or lines following the command. A single list element continues until the next .LIST ELEMENT command or an .END LIST command occurs. The list element can contain commands and flags.

To change the spacing between list elements, specify the number of spaces as parameter 1 to the .LIST command. For example, .LIST 0 places no spaces between list elements.

9.1.5.2 Bulleted Lists

If you do not want your list to be numbered, as it is by default, substitute another character for the numbers by specifying that character as the second parameter of the .LIST command. Enclose the character in quotation marks or apostrophes; the character does not have to be preceded by a comma if the first parameter is not specified. In the following example, the lowercase "o" gives the effect of an unfilled bullet:

```
.LIST "o"
.LIST ELEMENT;ferret
.LIST ELEMENT;mink
.LIST ELEMENT;rabbit
.LIST ELEMENT;sable
.LIST ELEMENT;raccoon
.END LIST
```

The preceding lines of text coded in DSR produce the following output:

```
o ferret

o mink

o rabbit

o sable

o raccoon
```

9.1.5.3 Nested Lists

You can nest one list within another as long as the nested list is entirely within one element of the outer list as follows:

```
.LIST 0
.LIST ELEMENT;German
.LIST ELEMENT;Russian
.LIST ELEMENT;Swedish
.LIST ELEMENT;Yugoslavian
.LIST 0,"o"
.LIST ELEMENT;Serbian
.LIST ELEMENT;Croatian
.LIST ELEMENT;Macedonian
.END LIST
.LIST ELEMENT;Turkish
.LIST ELEMENT;Scottish
.LIST ELEMENT;Irish
.END LIST
```

The preceding lines of text coded in DSR produce the following output:

```
1.  German
2.  Russian
3.  Swedish
4.  Yugoslavian
      o  Serbian
      o  Croatian
      o  Macedonian
5.  Turkish
6.  Scottish
7.  Irish
```

9.1.5.4 Lists Beginning with Letters and Roman Numerals

By default, DSR numbers lists with decimal numbers. You can, however, produce a list whose elements are preceded by uppercase or lowercase letters or Roman numerals by specifying the .DISPLAY ELEMENTS command. You must place the .DISPLAY ELEMENTS command between the .LIST command and the first .LIST ELEMENT command. The following example produces a numbered list using lowercase Roman numerals:

```
.LIST 0
.DISPLAY ELEMENTS RL
.LIST ELEMENT;tan
.LIST ELEMENT;beige
.LIST ELEMENT;rust
.LIST ELEMENT;brown
.END LIST
```

The preceding lines of text coded in DSR produce the following output:

```
  i.  tan
 ii.  beige
iii.  rust
 iv.  brown
```

9.1.6 Leaving Space on a Page

The text of a file usually starts on the fourth or fifth line from the top of a page, depending on the layout of the document. To leave extra space at the top of a page, specify an amount in a .FIGURE command (the .BLANK command does not work at the top of a page). To leave extra space in the middle of a page, use either the .FIGURE or .BLANK command. If you want a certain amount of space all on one page, use the .FIGURE or .FIGURE DEFERRED command. These commands insert the required space on the next page if it does not fit on the current page. While the .FIGURE command leaves the rest of the current page blank, the .FIGURE DEFERRED command fills the rest of the current page using the text and commands that follow the .FIGURE DEFERRED command.

The following example writes 40 blank lines to the current page if they will fit; otherwise, the 40 blank lines are placed at the top of the next page, and the current page is filled from the input lines following the .FIGURE DEFERRED line.

```
text filled and justified
.FIGURE DEFERRED 40
text filled and justified
```

You can also create space on a page by entering the .LITERAL command, pressing RETURN once for each empty line, and entering the .END LITERAL command. This technique allows you to see the amount of space you are creating. However, the space will be split if you cross page boundaries.

9.1.7 Formatting Notes

The .NOTE command narrows the left and right margins, inserts a blank line, writes a centered title, inserts another blank line, and writes the text that follows the .NOTE command. The .END NOTE command writes a blank line and restores the margin settings. The title defaults to the word NOTE.

The following example writes a note with the title CAUTION:

```
.NOTE CAUTION
Do not operate the machine outdoors in wet weather.
Do not operate the machine in a wet area indoors or outdoors.
Such actions may lead to a severe electrical shock.
.END NOTE
```

The preceding lines of text coded in DSR produce the following output:

```
                    CAUTION

     Do not operate the machine outdoors  in  wet  weather.
     Do  not  operate  the machine in a wet area indoors or
     outdoors.  Such  actions  may  lead  to  a  severe
     electrical shock.
```

9.1.8 Formatting Footnotes

The .FOOTNOTE command places the text following the command at the bottom of the page if enough room exists. If the entire footnote does not fit, the whole footnote is placed at the bottom of the next page. No automatic formatting occurs; you must determine how to format the footnote with DSR commands. In addition, no automatic footnote symbols are provided. The .END FOOTNOTE command ends the footnote and automatically restores any case, fill, justify, and margin settings you might have changed within the footnote.

The following example demonstrates the use of a footnote:

```
Press the START button firmly.
Release the START button as soon as the engine starts.(1)
.FOOTNOTE.BLANK.LEFT MARGIN +4.INDENT -4
(1)
If the engine does not crank, ensure that the battery cables
are firmly connected to the battery. Sometimes the cables are
disconnected for shipping.
.END FOOTNOTE
Push the choke in until you hear it click.
Pull the throttle about halfway down but do not let the
engine stall.
After about two minutes, push the choke all the way in
and pull the throttle all the way down.
.PARAGRAPH 0
Before engaging the drive train, ensure that you are in a
comfortable position to operate the machine.
Two seat controls are provided for your comfort.
```

The preceding lines of text coded in DSR produce the following output:

```
Press the START button firmly.  Release the START button as
soon as the engine starts.(1) Push the choke in until you
hear it click.  Pull the throttle about halfway down but do
not let the engine stall.  After about two minutes, push the
choke all the way in and pull the throttle all the way down.

Before engaging the drive train, ensure that you are in a
comfortable position to operate the machine.  Two seat
controls are provided for your comfort.

(1) If the engine does not crank, ensure that the battery
    cables are firmly connected to the battery. Sometimes
    the cables are disconnected for shipping.
```

9.1.9 Bolding and Underlining Text

The following steps describe how to bold a single character:

1. Turn the bold flag on. (The bold flag is off by default.)
2. In the text, precede the character to be bolded by the bold flag (an asterisk by default).
3. Turn the bold flag off to enable the use of the flag as a normal character.

The following example bolds the numbers 3 and 7:

```
.FLAGS BOLD
Follow route *3 to route *7.
.NOFLAGS BOLD
```

The following steps describe how to underline a single character:

1. Turn the underline flag on. (The underline flag is on by default.)
2. In the text, precede the character to be underlined by the underline flag (an ampersand by default).
3. Turn the underline flag off to enable the use of the flag as a normal character.

The following example underlines the letters A and B:

```
.FLAGS UNDERLINE
&A is for Amy and &B is for Basil.
.NOFLAGS UNDERLINE
```

The following steps describe how to bold or underline a block of text:

1. Turn on the uppercase and lowercase flags (they are on by default) in addition to the bold or underline flag.
2. Start the block of text with the uppercase flag (a circumflex by default) followed by the bold or underline flag.
3. End the block of text with the lowercase flag (a backslash by default) followed by the bold or underline flag.

The following example bolds a line of text:

```
.FLAGS BOLD
.FLAGS UPPERCASE
.FLAGS LOWERCASE
^*KEEP OFF THE GRASS, PLEASE\*
.NOFLAGS BOLD
.NOFLAGS UPPERCASE
.NOFLAGS LOWERCASE
```

9.2 Laying Out a Document

By default, DSR produces a document of consecutively numbered pages (that is, if you do not specify .LAYOUT, .CHAPTER, or .APPENDIX commands). On each page, the text area is the fourth line through the bottom line. Page numbers appear in the upper right corner as "Page 2," "Page 3," and so on, starting with page 2. Running heads (chapter names or other designated text) appear in the upper left corner.

You can adjust the position of page numbers and running heads with the .LAYOUT command. Layout codes 1, 2, and 3 center the page number at the bottom of the page and adjust the running heads as follows:

- Layout code 1 centers running heads at the top of the page.
- Layout code 2 moves the running heads to one upper corner or the other depending on whether the page number is odd or even.
- Layout code 3 puts the running heads in the upper left corner and puts the date in the upper right corner.

Specify the .LAYOUT command at the beginning of your file. The following command adjusts the layout to code 2:

```
.LAYOUT 2,3
```

9.2.1 Chapters and Appendixes

To divide a document into chapters, start each chapter with the .CHAPTER command. The title of the chapter must follow the command name on the same line. The lines following the .CHAPTER command are part of that chapter until you enter another .CHAPTER command or an .APPENDIX command.

By default, chapters are numbered consecutively within the document, beginning with Chapter 1. You can force the numbering of a chapter (for example, in order to place each chapter in a separate file) by preceding the .CHAPTER command with a .NUMBER CHAPTER command. The following example begins Chapter 2:

```
.NUMBER CHAPTER 2
 .CHAPTER starting procedures
```

The preceding lines of text coded in DSR produce the following output:

```
<12 blank lines>
                    CHAPTER 2

                STARTING PROCEDURES
```

DSR starts a chapter on a new page with 12 blank lines at the top of the page. The number of the chapter and the chapter title are centered on the page in uppercase characters. You can adjust the appearance of the chapter number with the .DISPLAY CHAPTER command.

The .APPENDIX, .NUMBER APPENDIX, and .DISPLAY APPENDIX commands work similarly to the chapter commands. However, by default, appendixes are lettered sequentially starting with Appendix A. The following example starts Appendix C:

```
.NUMBER APPENDIX C
 .APPENDIX connecting the battery
```

9.2.2 Sections

The .HEADER LEVEL command divides a document into sections and subsections identified by a decimal numbering scheme to a maximum depth of 6. The topmost section is header level 1. Each level is numbered sequentially starting with 1 unless a .NUMBER LEVEL command precedes the .HEADER LEVEL command. The following example shows a document with three sections:

```
text
.HEADER LEVEL 1 normal starting
text
.HEADER LEVEL 1 cold weather starting
text
.HEADER LEVEL 1 troubleshooting
text
```

The preceding lines of text coded in DSR produce the following output:

```
text
1  NORMAL STARTING
text
2  COLD WEATHER STARTING
text
3  TROUBLESHOOTING
text
```

Subsections are numbered within their respective higher-level sections as follows:

```
text
.HEADER LEVEL 1 normal starting
text
.HEADER LEVEL 2 cold engine
text
.HEADER LEVEL 2 warm engine
text
.HEADER LEVEL 1 cold weather starting
text
.HEADER LEVEL 2 above zero
text
.HEADER LEVEL 2 below zero
text
.HEADER LEVEL 1 troubleshooting
text
```

The preceding lines of text coded in DSR produce the following output:

```
text
1  NORMAL STARTING
text
1.1  Cold Engine
text
1.2  Warm Engine
text
2  COLD WEATHER STARTING
text
2.1  Above Zero
text
2.2  Below Zero
text
3  TROUBLESHOOTING
text
```

If the sections are within chapters or appendixes, the section number is prefixed by the chapter or appendix identifier and a decimal point as follows:

```
.NUMBER CHAPTER 2
.CHAPTER starting procedures
text
.HEADER LEVEL 1 normal starting
text
.HEADER LEVEL 1 cold weather starting
text
.HEADER LEVEL 1 troubleshooting
text
```

The preceding lines of text coded in DSR produce the following output:

```
the chapter heading and text
2.1  NORMAL STARTING
text
2.2  COLD WEATHER STARTING
text
2.3  TROUBLESHOOTING
text
```

You can force the numbering of a section with the .NUMBER LEVEL command. The following example forces the start of section 1.2:

```
.NUMBER LEVEL 1,2
.HEADER LEVEL 2 Warm Engine
```

You can change the appearance of section headers with the .STYLE HEADERS command. The default .STYLE HEADERS settings cause level 1 headers to be written in all uppercase and level 2 headers to be written with the initial letter of every word in uppercase. The following command changes the settings so that the headers below level 1 are written exactly as you type them:

```
.STYLE HEADERS 3,1,0,7,7,2,1,9,2
```

9.2.3 Running Heads

By default, chapter and appendix titles appear as the first line of running heads. If the document does not contain chapters or appendixes, no running heads appear. Running heads are always placed at the top of the page; their exact position can be changed with the .LAYOUT command.

To use level 1 header titles as the second line of running heads, enter the following commands at the start of your DSR file:

```
.SUBTITLE
.AUTOSUBTITLE
```

To use titles other than chapter and section titles as running heads, use the .TITLE and .SUBTITLE commands. The .TITLE command affects the first line of the running head; the .SUBTITLE command affects the second line of the running head.

The title specified by a .TITLE command remains in effect until you specify another .TITLE or .CHAPTER command. If you want to use a specified title in place of a chapter name, enter the .TITLE command immediately after the .CHAPTER command. The title specified by a .SUBTITLE command remains in effect until you specify another .SUBTITLE command or until a .HEADER LEVEL command occurs (if automatic subtitles are in effect).

9.2.4 Pagination

If the document is not divided into chapters, pagination is sequential throughout the document. If the document is chapter oriented, pagination is sequential throughout the document only if .LAYOUT 3 is in effect. Otherwise, pagination is sequential within each chapter and appendix; the page number starts with the chapter or appendix identifier and a hyphen.

You can suspend the numbering of pages with the .NO NUMBER command (unless .LAYOUT 3 is in effect). Create a document that is not paged (no running heads, no page numbers) by entering the command .NO PAGING.

9.3 Processing DSR Files

Enter the RUNOFF command to process a DSR file. Specify the name of the DSR file as the parameter; the file type defaults to RNO. The following example processes a file named EXAMPLE.RNO in your default directory.

```
$ RUNOFF EXAMPLE
```

If you do not specify the /OUTPUT qualifier, the RUNOFF command produces an output file with the same name as the input file and a file type of MEM. The preceding example produces an output file named EXAMPLE.MEM. The following example produces an output file named EXAMPLE.MEM from a DSR file named TEMPLATE.RNO:

```
$ RUNOFF/OUTPUT=EXAMPLE TEMPLATE
```

See the Reference Section for a complete description of the RUNOFF command and its qualifiers.

9.3.1 Producing a Table of Contents

The table of contents you produce can display chapter titles and numbers, header levels, and appendix titles and letters. To produce a table of contents, do the following:

1. Enter the command RUNOFF/INTERMEDIATE, specifying the RNO file as the parameter. The name of the intermediate file produced is the same as the name of the DSR file with a file type of BRN, unless you specify a different name. You will also get the usual output (MEM) file; you can specify /NOOUTPUT if you do not want the MEM file.

2. Enter the command RUNOFF/CONTENTS, specifying the intermediate file as the parameter. This command produces an unformatted table of contents file with the same name as the input file but with a file type of RNT.

3. Enter the RUNOFF command, specifying the RNT file as the parameter. You must specify the file type. This command produces a formatted table of contents file with a file type of MEC.

The following example processes a file named OPER.RNO. It produces an output file named OPER.MEM and a table of contents named OPER.MEC:

```
$ RUNOFF/INTERMEDIATE OPER
$ RUNOFF/CONTENTS OPER
$ RUNOFF OPER.RNT
```

To produce a table of contents from more than one file, you must concatenate the intermediate files when you enter the RUNOFF/CONTENTS command. (You cannot use wildcard characters.) The following example produces output files and a single table of contents from three DSR files:

```
$ RUNOFF/INTERMEDIATE OPER1
$ RUNOFF/INTERMEDIATE OPER2
$ RUNOFF/INTERMEDIATE OPER3
$ RUNOFF/CONTENTS/OUTPUT=OPER OPER1+OPER2+OPER3
$ RUNOFF OPER.RNT
```

The table of contents is based on the .CHAPTER, .APPENDIX, and .HEADER LEVEL commands in your DSR file. You can control the formatting to some extent with the qualifiers to the RUNOFF/CONTENTS command. You can write additional information to the table of contents with the command .SEND TOC.

See the description of the DCL command RUNOFF/CONTENTS and the DSR commands in the Reference Section for more information about producing a table of contents with DSR.

9.3.2 Producing an Index

To create an index, you enter .INDEX and .ENTRY commands throughout your DSR file. (You can also use the index flag to index a word of text.)

The .INDEX command names an item to be placed in the index. Position the .INDEX command as close as possible to the text being indexed. The item appears in the index followed by the number of the page on which it was written to the formatted text (MEM) file. The following example makes index entries for each section:

```
text
.HEADER LEVEL 1 Normal Starting
.INDEX Normal starting
text
.HEADER LEVEL 1 Cold Weather Starting
.INDEX Cold weather starting
text
.HEADER LEVEL 1 Troubleshooting
.INDEX Troubleshooting
text
```

The index entries would appear as follows:

```
Cold weather starting, 2-2
Normal starting, 2-1
Troubleshooting, 2-3
```

Use the subindex flag (by default a right angle bracket) to indicate subentries in the index. A subentry is listed under the higher-level item in the index and has its own page number. The subindex flag must be turned on. The following example produces one index entry for "Starting" under which are listed the two subentries "normal" and "cold weather":

```
text
.HEADER LEVEL 1 Normal Starting
.FLAGS SUBINDEX
.INDEX Normal starting
.INDEX Starting>normal
.NOFLAGS SUBINDEX
text
.HEADER LEVEL 1 Cold Weather Starting
.FLAGS SUBINDEX
.INDEX Cold weather starting
.INDEX Starting>cold weather
.NOFLAGS SUBINDEX
text
.HEADER LEVEL 1 Troubleshooting
.INDEX Troubleshooting
text
```

The index entry for "Starting" would appear as follows:

```
Starting
  cold weather, 2-2
  normal, 2-1
```

You can make an entry without a page number in the index with the .ENTRY command. Usually these index entries are used for cross references. The following example produces an index entry for "Weather," under which the subentry "see cold weather" appears without a page number:

```
.FLAGS SUBINDEX
.ENTRY Weather>see cold weather
```

The index entry would appear as follows:

```
Weather
  see cold weather
```

After you enter the index commands to your file, you are ready to run the indexing program. To produce an index, do the following:

1. Enter the command RUNOFF/INTERMEDIATE, specifying your RNO file as the parameter. The name of the intermediate file produced is the same as the name of the DSR file but with a file type of BRN, unless you specify a different name. You will also get the usual output (MEM) file; you can specify /NOOUTPUT if you do not want the MEM file.

2. Enter the command RUNOFF/INDEX, specifying the intermediate file as the parameter. This command produces an unformatted index file with the same name as the input file but with a file type of RNX.

3. Enter the RUNOFF command, specifying the RNX file as the parameter. You must specify the file type. This command produces a formatted index file with a file type of MEX.

The following example processes a file named OPER.RNO. It produces an output file named OPER.MEM, a table of contents named OPER.MEC, and an index named OPER.MEX.

```
$ RUNOFF/INTERMEDIATE OPER
$ RUNOFF/CONTENTS OPER
$ RUNOFF/INDEX OPER
$ RUNOFF OPER.RNT
$ RUNOFF OPER.RNX
```

To produce an index from more than one file, you must concatenate the intermediate files when you enter the RUNOFF/INDEX command. (You cannot use wildcard characters.) The following example produces formatted text files, a single table of contents, and a single index from three DSR files:

```
$ RUNOFF/INTERMEDIATE OPER1
$ RUNOFF/INTERMEDIATE OPER2
$ RUNOFF/INTERMEDIATE OPER3
$ RUNOFF/CONTENTS/OUTPUT=OPER OPER1+OPER2+OPER3
$ RUNOFF/INDEX/OUTPUT=OPER OPER1+OPER2+OPER3
$ RUNOFF OPER.RNT
$ RUNOFF OPER.RNX
```

9.3.3 Printing Output Files

The following are guidelines for printing nonlaser-output files produced by DSR:

- Copying files—You can copy a file (with the COPY command) to a printer, but you may occasionally lose a form feed. You lose a form feed when the size of an output page equals 66 or the value that SYS$LP_LINES had at the time the file was created if SYS$LP_LINES was defined as a logical name. (Note that the current value of SYS$LP_LINES has no effect on the output file after it is created.)
- Generating automatic form feeds—In general, use the default for the DCL command PRINT (PRINT/FEED) to generate form feeds automatically when printing nonlaser files. However, you may occasionally generate a blank page. More precisely, you generate a blank page when an output page equals 66 or the value that SYS$LP_LINES had at the time the file was created if SYS$LP_LINES was defined as a logical name. (Note that the current value of SYS$LP_LINES has no effect on the output file.) If you specify PRINT/NOFEED, you eliminate the chance of a blank page, but you take the chance of losing pages under the same circumstances as a COPY operation.

The following example shows a laser printer setup in a command procedure. *Dsr$ln01* is the name you assign the laser printer form. *Lpb0* is the name of the printer.

```
$ ! Define form for dsr output on ln01 laser printer
$
$ DEFINE/FORM dsr$ln01 /MARGIN=(BOTTOM=0) -
                       /NOWRAP -
                       /NOTRUNCATE -
                       /STOCK=DEFAULT -
                       /DESCRIPTION="dsr ln01 form definition"
$
$ ! Set up ln01 laser printer for dsr output
$
$ SET PRINTER lpb0     /NOTRUNCATE -
                       /NOWRAP -
                       /TAB -
                       /PRINTALL -
                       /FF -
                       /NOCR
```

The print command should specify /NOFEED, the name of the form, and the name of the queue. You should equate this command to a global symbol in your login file or in the system login file. The following example makes LNPRINT a global symbol that prints LN01 files:

```
$ ! Set up lnprint as print command for ln01 laser printer
$
$ LNP*RINT == "PRINT/NOFEED/FORM=dsr$ln01/QUEUE=ln01_queue"
```

GENERAL USER'S REFERENCE

DCL Commands

This section describes each DCL command and lexical function. The commands are listed in alphabetical order, with the command name appearing at the top of every page. The lexical functions are grouped alphabetically under "Lexical Functions" (after the JOB command description); the name of the lexical function appears at the top of each page.

= (Assignment Statement)

Defines a symbolic name for a character string or integer value.

format

symbol-name =[=] expression

parameters

symbol-name

Specifies a 1 to 255 character alphanumeric string name for the symbol. The name can contain any alphanumeric characters from the DEC Multinational Character Set, the underscore (_), and the dollar sign ($). However, the name must begin *only* with an alphabetic character, an underscore, or a dollar sign. Using one equal sign (=) places the symbol name in the local symbol table for the current command level. Using two equal signs (==) places the symbol name in the global symbol table.

expression

Names the value on the right-hand side of an assignment statement. Can consist of a character string, an integer, a symbol name, a lexical function, or a combination of these entities. The components of the expression are evaluated, and the result is assigned to the symbol. All literal character strings must be enclosed in quotation marks. If the expression contains a symbol, the expression is evaluated using the symbol's value.

example

```
$ LIST == "DIRECTORY"
```

The assignment statement in this example assigns the user-defined synonym LIST as a global symbol definition for the DCL command DIRECTORY.

:= (String Assignment)

Defines a symbolic name for a character string value.

format

symbol-name :=[=] string

parameters

symbol-name

Specifies a 1 to 255-character string name for the symbol. The name can contain any alphanumeric characters from the DEC Multinational Character Set, the underscore, and the dollar sign. However, the name must begin *only* with an alphabetic character, an underscore (_), or a dollar sign ($). Using one equal sign (:=) places the symbol name in the local symbol table for the current command level. Using two equal signs (:==) places the symbol name in the global symbol table.

string

Names the character string value to be equated to the symbol. The string can contain any alphanumeric or special characters. String values are automatically converted to uppercase. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space. To prohibit uppercase conversion and retain required space and tab characters in a string, place quotation marks around the string.

example

```
$ TIME := SHOW TIME
$ TIME
  15-APR-1988 11:55:44
```

In this example, the symbol TIME is equated to the command string SHOW TIME. Because the symbol name appears as the first word in a command string, the command interpreter automatically substitutes it with its string value and executes the command SHOW TIME.

@ (Execute Procedure)

Executes a command procedure or requests the command interpreter to read subsequent command input from a specific file or device.

format

@ *file-spec [p1 [p2 [... p8]]]*

parameters

file-spec

Specifies either the input device or file for the preceding command, or the command procedure to be executed. The default file type is COM. Wildcard characters are not allowed in the file specification.

p1 [p2 [... p8]]

Specifies from one to eight optional parameters to pass to the command procedure. The symbols (P1, P2, . . . P8) are assigned character string values in the order of entry. The symbols are local to the specified command procedure. Separate each parameter with one or more blanks. Use two consecutive quotation marks ("") to specify a null parameter.

qualifier

/OUTPUT=file-spec

The name of the file to which the command procedure output is written. By default, the output is written to the current SYS$OUTPUT device. The default output file type is LIS. Wildcard characters are not allowed in the output file specification. System responses and error messages are written to SYS$COMMAND as well as to the specified file. The /OUTPUT qualifier must immediately follow the file specification of the command procedure; otherwise, the qualifier is interpreted as a parameter to pass to the command procedure.

example

```
$ CREATE DOFOR.COM
$ ON WARNING THEN EXIT
$ IF P1.EQS."" THEN INQUIRE P1 FILE
$ FORTRAN/LIST 'P1'
$ LINK 'P1'
$ RUN 'P1'
$ PRINT 'P1'
CTRL/Z
$ @DOFOR AVERAGE
```

This example shows a command procedure, named DOFOR.COM, that executes the FORTRAN, LINK, and RUN commands to compile, link, and execute a program. The ON command requests that the procedure not continue if any of the commands result in warnings or errors.

When you execute DOFOR.COM, you can pass the file specification of the FORTRAN program as the parameter P1. If you do not specify a value for P1 when you execute the procedure, the INQUIRE command issues a prompting message to the terminal and equates what you enter with the symbol P1. In this example, the file name AVERAGE is assigned to P1. The file type is not included because the commands FORTRAN, LINK, RUN, and PRINT provide default file types.

ACCOUNTING

Invokes the Accounting Utility to collect, record, and report accounting data. For more information about the Accounting Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ALLOCATE

Provides your process with exclusive access to a device until you deallocate the device or terminate your process. Optionally associates a logical name with the device.

format

ALLOCATE *device-name[:][,...] [logical-name[:]]*

parameters

device-name[:][,...]
Specifies the name of a physical device or a logical name that translates to the name of a physical device. The device name can be generic: if no controller or unit number is specified, any device that satisfies the specified part of the name is allocated. If more than one device is specified, the first available device is allocated.

logical-name
Specifies a character string of 1 through 255 characters. Enclose the string in quotation marks (") if it contains blanks. Trailing colons are not used. The name becomes a process logical name with the device name as the equivalence name. The logical name remains defined until it is explicitly deleted or your process terminates.

qualifiers

/GENERIC
/NOGENERIC (default)
Indicates that the first parameter is a device *type* rather than a device *name*. Example device types are RX50, RD52, TK50, RC25, RCF25, RL02. The first free, nonallocated device of the specified name and type is allocated.

/LOG (default)
/NOLOG

Displays a message indicating the name of the device allocated. If the operation specifies a logical name that is currently assigned to another device, displays the superseded value.

example

```
$ ALLOCATE /GENERIC RX50 ACCOUNTS
```

The ALLOCATE command in this example allocates the first free floppy disk drive and makes its name equivalent to the process logical name ACCOUNTS.

ANALYZE/CRASH_DUMP

Invokes the System Dump Analyzer Utility (SDA) for analysis of a system dump file. The /CRASH_DUMP qualifier is required.

format

ANALYZE/CRASH_DUMP *file-spec*

ANALYZE/DISK_STRUCTURE

Invokes the Analyze/Disk_Structure Utility to do the following:

- Check the readability and validity of Files–11 Structure Level 1 and Files–11 Structure Level 2 disk volumes
- Report errors and inconsistencies

The /DISK_STRUCTURE qualifier is required. For more information about the Analyze/Disk_Structure Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/ERROR_LOG

Invokes the Errorlog Report Formatter (ERF) to report selectively the contents of an error log file. The /ERROR_LOG qualifier is required. For more information about the Error Log Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/IMAGE

Analyzes the contents of an executable image file or a shareable image file and checks for obvious errors in the image file. See the description of the linker for general information about image files.

format

ANALYZE/IMAGE *file-spec [,...]*

parameter

file-spec[,...]
Specifies the image files you want analyzed (default file type is EXE.) Use commas or plus signs to separate file specifications. Wildcard characters are allowed.

qualifiers

/FIXUP_SECTION
Positional Qualifier. If you want the analysis to include the fixup section of all image files in the parameter list, insert the /FIX_UP qualifier immediately following the /IMAGE qualifier. If you want the analysis to include fixup sections selectively, insert the /FIX_UP qualifier immediately following the selected file specification(s).

/GST
Positional Qualifier. This qualifier is valid only for shareable images. If you want the analysis to include the global symbol table records of all image files in the parameter list, insert the /GST qualifier immediately following the /IMAGE qualifier. If you want the analysis to include global symbol table records selectively, insert the /GST qualifier immediately following the selected file specification(s).

/HEADER
Positional Qualifier. Specifies that the analysis should include only header items and image section descriptions, unless the command explicitly specifies other information.

/INTERACTIVE
/NOINTERACTIVE (default)
Specifies whether or not the analysis is interactive.

/OUTPUT=file-spec
Directs the output of the image analysis (default is SYS$OUTPUT.) No wildcard characters are allowed.

/PATCH_TEXT

Positional Qualifier. If you want the analysis to include the patch text records for each image file in the parameter list, insert the /PATCH_TEXT qualifier immediately following the /IMAGE qualifier. If you want the analysis to include patch text records selectively, insert the /PATCH_TEXT qualifier immediately following the selected file specification(s).

example

```
$ ANALYZE/IMAGE/OUTPUT=LIALPHEX/FIXUP_SECTION/PATCH_TEXT  LINEDT, ALPHA
```

The ANALYZE/IMAGE command in this example stores a description and an error analysis of the fixup sections and patch text records of LINEDT.EXE and ALPHA.EXE in file LIALPHEX.ANL. The output is directed to the file LIALPHEX.ANL.

ANALYZE/MEDIA

Invokes the Bad Block Locator Utility (BAD), which analyzes block-addressable devices and records the location of blocks that cannot reliably store data. For more information about the Bad Block Locator Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/OBJECT

Analyzes the contents of an object file and checks for any obvious errors.

format

ANALYZE/OBJECT *file-spec[,...]*

parameter

file-spec[,...]

Specifies the object files or object module libraries you want analyzed (default file type is OBJ). Use commas or plus signs to separate file specifications. Wildcard characters are allowed.

qualifiers

/DBG

Positional qualifier. If you want the analysis to include debugger information for all files in the parameter list, insert the /DBG qualifier immediately following the /OBJECT qualifier. If you want the analysis to include debugger information selectively, insert the /DBG qualifier immediately following the selected file specification(s).

/EOM

Positional qualifier. Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this to apply to all files in the parameter list, insert the /EOM qualifier immediately following the /OBJECT qualifier. To make this applicable selectively, insert the /EOM qualifier immediately following the selected file specification(s).

/GSD

Positional qualifier. If you want the analysis to include global symbol directory records for each file in the parameter list, specify /GSD immediately following the /OBJECT qualifier. If you want the analysis to include global symbol directory records selectively, insert the /GSD qualifier immediately following the selected file specification(s).

/INCLUDE[=(module[,...])]

When the specified file is an object module library, use this qualifier to list selected object modules within the library for analysis. If you omit the list or specify an asterisk, all modules are analyzed.

/INTERACTIVE
/NOINTERACTIVE (default)

Controls whether the analysis occurs interactively.

/LNK

Positional qualifier. If you want the analysis to include link option specification records for each file in the parameter list, specify /LNK immediately following the /OBJECT qualifier. If you want the analysis to include link option specification records selectively, insert the /LNK qualifier immediately following the selected file specification(s).

/MHD

Positional qualifier. Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this to apply to all files in the parameter list, insert the /MHD qualifier immediately following the /OBJECT qualifier. To make this applicable selectively, insert the /MHD qualifier immediately following the selected file specification(s).

/OUTPUT[=file-spec]

Directs the output of the object analysis (default is SYS$OUTPUT). No wildcard characters are allowed in the file specification.

/TBT

Positional qualifier. If you want the analysis to include traceback records for each file in the parameter list, specify /TBT immediately following the /OBJECT qualifier. If you want the analysis to include traceback records selectively, insert the /TBT qualifier immediately following the selected file specification(s).

/TIR

Positional qualifier. If you want the analysis to include text information and relocation records for each file in the parameter list, specify /TIR immediately following the /OBJECT qualifier. If you want the analysis to include text information and relocation records selectively, insert the /TIR qualifier immediately following the selected file specification(s).

example

```
$ ANALYZE/OBJECT/OUTPUT=LIOBJ/DBG  LINEDT
```

In this example, the ANALYZE/OBJECT command analyzes only the debugger information records of the file LINEDT.OBJ. Output is to the file LIOBJ.ANL.

ANALYZE/PROCESS_DUMP

Invokes the VMS Debugger for analysis of a process dump file that was created when an image failed during execution (use the /DUMP qualifier with the RUN or SET PROCESS commands to generate a dump file).

Requires read (R) access to the dump file.

format

ANALYZE/PROCESS_DUMP *dump-file*

parameter

dump-file

Specifies the dump file to be analyzed with the debugger.

qualifiers

/FULL

Displays all known information about the failing process.

/IMAGE=image-name
/NOIMAGE

Specifies the image whose symbols are to be used in analyzing the dump. If you use the /NOIMAGE qualifier, no symbols are taken from any image. By default, symbols are taken from the image with the same name as the image that was running at the time of the dump.

/INTERACTIVE
/NOINTERACTIVE (default)

Causes the display of information to pause when your terminal screen is filled. Press RETURN to display additional information. By default, the display is continuous.

/MISCELLANEOUS

Displays all the miscellaneous information in the dump.

/OUTPUT=file-spec

Writes the information to the specified file. By default, the information is written to the current SYS$OUTPUT device.

/RELOCATION

Displays the addresses to which data structures saved in the dump are mapped in P0 space.

example

```
$ ANALYZE/PROCESS/FULL ZIPLIST

 R0 = 00018292  R1 = 8013DE20  R2 = 7FFE6A40   R3 = 7FFE6A98
 R4 = 8013DE20  R5 = 00000000  R6 = 7FFE7B9A   R7 = 0000F000
 R8 = 00000000  R9 = 00000000  R10 = 00000000  R11 = 00000000
 SP = 7FFAEF44  AP = 7FFAEF48  FP  = 7FFAEF84
 FREE_P0_VA  00001600    FREE_P1_VA  7FFAC600
 Active ASTs  00         Enabled ASTs 0F
 Current Privileges  FFFFFF80  1010C100
 Event Flags  00000000  E0000000
 Buffered I/O count/limit 6/6
 Direct I/O count/limit   6/6
 File count/limit         27/30
 Process count/limit      0/0
 Timer queue count/limit  10/10
 AST count/limit          6/6
 Enqueue count/limit      30/30
 Buffered I/O total 7      Direct I/O total 18
 Link Date  27-DEC-1988 15:02:00.48   Patch Date  17-NOV-1988 00:01:53.71
 ECO Level  0030008C  00540040  00000000  34303230
 Kernel stack 00000000 pages at 00000000 moved to 00000000
 Exec stack 00000000 pages at 00000000 moved to 00000000
 Vector page 00000001 page at 7FFEFE00 moved to 00001600
 PIO (RMS) area 00000005 pages at 7FFE1200 moved to 00001800
 Image activator context 00000001 page at 7FFE3400 moved to 00002200
 User writeable context 0000000A pages at 7FFE1C00 moved to 00002400
Creating a subprocess
        VAX DEBUG Version X5.0-2
DBG>
```

This example shows the output of the ANALYZE/PROCESS command when used with the /FULL qualifier. The file specified, ZIPLIST, contains the dump of a process that encountered a fatal error. The DBG> prompt indicates that the debugger is ready to accept commands.

ANALYZE/RMS_FILE

Invokes the Analyze/RMS_File Utility (ANALYZE/RMS_FILE) to inspect and analyze the internal structure of a VMS RMS file. The /RMS_FILE qualifier is required.

format

ANALYZE/RMS_FILE *file-spec[,...]*

ANALYZE/SYSTEM

Invokes the System Dump Analyzer (SDA) for analysis of the running system. The /SYSTEM qualifier is required.

format

ANALYZE/SYSTEM

APPEND

Adds the contents of one or more specified input files to the end of the specified output file.

format

APPEND *input-file-spec[,...] output-file-spec*

parameters

input-file-spec[,...]
Specifies the names of one or more input files to be appended. Multiple input files are appended to the output file in the order specified. If you specify more than one input file, separate multiple file specifications with either commas or plus signs. You can use wildcard characters in the input file specifications.

output-file-spec
Specifies the name of the file to which the input files will be appended. You must specify at least one field in the output file specification. If you do not specify a device or directory, the APPEND command uses the current default device and directory. Other unspecified fields default to the corresponding fields of the first input file specification.

qualifiers

/ALLOCATION=number-of-blocks

Output-file-spec qualifier. Forces the initial allocation of the output file to the specified number of 512-byte blocks. If you do not specify the /ALLOCATION qualifier, the initial allocation of the output file is determined by the size of the input file. Relevant only with the /NEW_VERSION qualifier.

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each APPEND operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CONTIGUOUS
/NOCONTIGUOUS
Output-file-spec qualifier. Specifies that the output file must occupy physically contiguous disk blocks. By default, the APPEND command creates an output file in the same format as the corresponding input file and does not report an error if not enough space exists for a contiguous allocation. Relevant only with the /NEW_VERSION qualifier.

/CREATED (default)
Modifies the time value specified with the /BEFORE or /SINCE qualifiers. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])
Excludes the specified files from the append operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXTENSION=number-of-blocks
Output-file-spec qualifier. Specifies the number of blocks to be added to the output file each time the file is extended. When you specify /EXTENSION, the /NEW_VERSION qualifier is assumed and need not be typed on the command line. Relevant only with the /NEW_VERSION qualifier.

/LOG
/NOLOG (default)
Controls whether the APPEND command displays the file specifications of each file appended. If /LOG is specified, displays the file specifications of the input and output files as well as the number of blocks or records appended after each append operation.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED,

and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/NEW_VERSION
/NONEW_VERSION (default)
Output-file-spec qualifier. Controls whether the APPEND command creates a new output file if the specified output file does not exist. If the specified output file does not already exist, use the /NEW_VERSION qualifier to create a new output file. If the output file does exist, the /NEW_VERSION qualifier is ignored and the input file is appended to the output file.

/PROTECTION=(code)
Output-file-spec qualifier. Specifies protection for the output file. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (read), W (write), E (execute), or D (delete). The default protection, including any protection attributes not specified, is that of the existing output file. If no output file exists, the current default protection applies. Relevant only with the /NEW_VERSION qualifier.

/READ_CHECK
/NOREAD_CHECK (default)
Input-file-spec qualifier. Reads each record in the input files twice to verify that it has been read correctly.

/SINCE[=time]
Selects for the append operation only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/WRITE_CHECK
/NOWRITE_CHECK (default)
Output-file-spec qualifier. Reads each record in the output file after the record is written to verify that it was appended successfully and that the output file can subsequently be read without error.

example

```
$ APPEND/NEW_VERSION/LOG *.TXT   MEM.SUM
%APPEND-I-CREATED, USE$:[MAL]MEM.SUM;1 created
%APPEND-S-COPIED, USE$:[MAL]A.TXT;2 copied to USE$:[MAL]MEM.SUM;1 (1 block)
%APPEND-S-APPENDED, USE$:[MAL]B.TXT;3 appended to USE$:[MAL]MEM.SUM;1 (3 records)
%APPEND-S-APPENDED, USE$:[MAL]G.TXT;7 appended to USE$:[MAL]MEM.SUM;1 (51 records)
```

The APPEND command appends all files with file types of TXT to a file named MEM.SUM. The /LOG qualifier requests a display of the specifications of each input file appended. If the file MEM.SUM does not exist, the APPEND command creates it, as the output shows. The number of blocks or records shown in the output refers to the source file and not to the target file total.

ASSIGN

Creates a logical name and assigns an equivalence string, or a list of strings, to the specified logical name. If you specify an existing logical name, the new equivalence name replaces the existing equivalence name.

format

ASSIGN *equivalence-name[,...] logical-name[:]*

parameters

equivalence-name[,...]

Specifies a character string of 1 to 255 characters. Defines the equivalence name, usually a file specification, device name, or other logical name, to be associated with the logical name in the specified logical name table. If the string contains other than uppercase alphanumeric, dollar sign, or underscore characters, enclose it in quotation marks (″). Use two consecutive quotation marks (″″) to denote an actual quotation mark. Specifying more than one equivalence name for a logical name creates a search list.

logical-name

Specifies the logical name string, which is a character string containing up to 255 characters. You choose a logical name to represent the equivalence name in the specified logical name table. If the string contains other than uppercase alphanumeric, dollar sign, or underscore characters, enclose it in quotation marks (″). Use two consecutive quotation marks (″″) to denote an actual quotation mark. If you terminate the logical-name parameter with a colon, the system removes the colon before placing the name in a logical name table. (This differs from the DEFINE command, which saves the colon.) If the logical name is to be entered into the process directory (LNM$PROCESS_DIRECTORY) or system directory (LNM$SYSTEM_DIRECTORY) logical name tables, then the name may only have from 1 to 31 alphanumeric characters (including the dollar sign and underscore). By default, the logical name is placed in the process logical name table.

qualifiers

/EXECUTIVE_MODE

Requires SYSNAM privilege. Specifies the mode of the logical name. If you specify executive mode, but do not have SYSNAM privilege, the qualifier is ignored and a supervisor mode logical name is created. The mode of the logical name must be the same as or external to (less privileged than) the mode of the table in which you are placing the name.

/GROUP

Requires SYSPRV or GRPNAM privilege. Places the logical name in the group logical name table. Other users who have the same group number in their user identification codes (UICs) can access the logical name. The /GROUP qualifier is synonymous with /TABLE=LNM$GROUP.

/JOB

Requires SYSPRV or GRPNAM privilege. Places the logical name in the jobwide logical name table. All processes within the same job tree as the process creating the logical name can access the logical name. The /JOB qualifier is synonymous with /TABLE=LNM$JOB.

/LOG (default)
/NOLOG

Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[=(keyword[,...])]

Specifies the attributes for a logical name. By default, no attributes are set. You can specify the following keywords for attributes:

CONFINE	Does not copy the logical name into a spawned subprocess; relevant only for logical names in a private table.
NO_ALIAS	Prohibits creation of logical names with the same name in an outer (less privileged) access mode within the specified table. If another logical name with the same name and an outer access mode already exists in this table, the name is deleted.

/PROCESS (default)

Places the logical name in the process logical name table. The /PROCESS qualifier is synonymous with /TABLE=LNM$PROCESS.

/SUPERVISOR_MODE (default)

Creates a supervisor mode logical name in the specified table.

/SYSTEM

Requires SYSNAM or SYSPRV privilege. Places the logical name in the system logical name table. All system users can access the logical name. The /SYSTEM qualifier is synonymous with /TABLE=LNM$SYSTEM.

/TABLE=name

Requires WRITE (W) access to the table if the table is shareable. Specifies the logical name table in which the logical name is to be entered. You can use the /TABLE qualifier to specify a user-defined logical name table (created with the CREATE/NAME_TABLE command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables. If you specify the table name using a logical name that has more than one translation, the logical name is placed in the first table found. If you do not explicitly specify the /TABLE qualifier, the default is /TABLE=LNM$PROCESS (or /PROCESS).

/TRANSLATION_ATTRIBUTES[=(keyword[,...])]

Equivalence-name qualifier. Specifies attributes of the equivalence-name parameter. Possible keywords are as follows:

CONCEALED Indicates that the equivalence string is the name of a concealed device.

TERMINAL Indicates that the equivalence string should not be translated iteratively; logical name translation should terminate with the current equivalence string.

/USER_MODE

Creates a user mode logical name in the specified table.

If you specify a user mode logical name in the process logical name table, that logical name is used for the execution of a single image only; user mode entries are deleted from the logical name table when any image executing in the process exits; that is, after any DCL command or user program that executes an image completes execution.

example

```
$ ASSIGN XXX1:[CHARLES]  CHARLIE
$ PRINT  CHARLIE:TEST.DAT
Job 274 entered on queue SYS$PRINT
```

The ASSIGN command in this example associates the logical name CHARLIE with the directory name [CHARLES] on the disk XXX1. Subsequent references to the logical name CHARLIE result in the correspondence between the logical name CHARLIE and the disk and directory specified. The PRINT command queues a copy of the file XXX1:[CHARLES]TEST.DAT to the system printer.

ASSIGN/MERGE

Removes all jobs from one queue and merges them into another existing queue. Does not affect jobs that are executing.

Requires OPER privilege or EXECUTE access to both queues.

format

ASSIGN/MERGE *target-queue[:] source-queue[:]*

parameters

target-queue[:]

Specifies the name of the queue into which the jobs are being merged.

source-queue[:]

Specifies the name of the queue from which the jobs are being removed.

example

```
$ STOP/QUEUE/NEXT LPB0
$ STOP/QUEUE/REQUEUE=LPA0 LPB0
$ ASSIGN/MERGE LPA0 LPB0
```

In this example, the STOP/QUEUE/NEXT command prevents another job from executing on queue LPB0. The STOP/QUEUE/REQUEUE command requeues the current job running on LPB0 to the target queue LPA0. The ASSIGN/MERGE command removes the remaining jobs from the LPB0 printer queue and places them in the LPA0 printer queue.

ASSIGN/QUEUE

Assigns, or redirects, a logical queue to a single execution queue. ASSIGN /QUEUE can be used only with printer or terminal queues.

Requires OPER privilege or EXECUTE access to both queues.

format

ASSIGN/QUEUE *queue-name[:] logical-queue-name[:]*

parameters

queue-name[:]
Name of the execution queue. The queue cannot be a logical queue, a generic queue, or a batch queue.

logical-queue-name[:]
Name of the logical queue.

example

```
$ INITIALIZE/QUEUE/DEFAULT=FLAG=ONE/START LPA0
$ INITIALIZE/QUEUE TEST_QUEUE
$ ASSIGN/QUEUE LPA0 TEST_QUEUE
$ START/QUEUE TEST_QUEUE
```

This example first initializes and starts the printer queue LPA0. The LPA0 queue is set to have a flag page precede each job. The second INITIALIZE /QUEUE command creates the logical queue TEST_QUEUE. The ASSIGN /QUEUE command assigns the logical queue TEST_QUEUE to the printer queue LPA0. The START/QUEUE command starts the logical queue.

ATTACH

Transfers control from your current process (which then hibernates) to the specified process.

The ATTACH and SPAWN commands cannot be used if your terminal has an associated mailbox.

format

ATTACH *[process-name]*

parameter

process-name
Specifies the name of a parent process or spawned subprocess to which control passes. The process must already exist, be part of your current job, and share the same input stream as your current process. However, the process cannot be your current process or a subprocess created with the /NOWAIT qualifier. The process-name parameter is incompatible with the /IDENTIFICATION qualifier.

qualifier

/IDENTIFICATION=pid
Specifies the process identification (PID) of the process to which terminal control will be transferred. Leading zeros can be omitted. The /IDENTIFICATION qualifier is incompatible with the process-name parameter.

example

```
$ATTACH JONES_2
```

Transfers the terminal's control to the subprocess JONES_2.

BACKUP

Invokes the Backup Utility (BACKUP) to perform one of the following BACKUP operations:

- Make copies of disk files.
- Save disk files as data in a file created by BACKUP on disk or magnetic tape. (Files created by BACKUP are called save sets.)
- Restore disk files from a BACKUP save set.
- Compare disk files or files in a BACKUP save set with other disk files.

- List information about files in a BACKUP save set to an output device or file.

Note that standalone BACKUP cannot be invoked this way, but must be bootstrapped in order to run. For more information about the Backup Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

CALL

Tranfers control to a labeled subroutine within a command procedure. The CALL command creates a new procedure level as does the @ (execute procedure) command.

format

CALL *label [p1[p2[... p8]]]*

parameters

label

Specifies a 1- to 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the CALL command is executed, control passes to the command following the specified label.

The label can precede or follow the CALL statement in the current command procedure. A label in a command procedure must be terminated with a colon.

All labels are procedure level dependent except for those labels that define subroutine entry points. The subroutine entry point labels are local to the current command procedure file level and must be unique.

p1 [p2 [... p8]]

Specifies from one to eight optional parameters to pass to the command procedure. Use two consecutive quotation marks ("") to specify a null parameter. The parameters assign character string values to the symbols named P1, P2, and so on in the order of entry, to a maximum of eight. The symbols are local to the specified command procedure. Separate each parameter with one or more blanks.

qualifier

/OUTPUT=file-spec

Writes all output to the file or device specified. By default, the output is written to the current SYS$OUTPUT device and the output file type is LIS. System responses and error messages are written to SYS$COMMAND as well as to the specified file. If you specify /OUTPUT, the qualifier must immediately follow the CALL command. No wildcard characters are allowed in the output file specification.

example

```
$
$! CALL.COM
$
$! Define subroutine SUB1
$!
$ SUB1: SUBROUTINE
    .
    .
    .
$ CALL SUB2 !Invoke SUB2 from within SUB1
    .
    .
    .
$ @FILE  !Invoke another procedure command file
    .
    .
    .
$ EXIT
$ ENDSUBROUTINE !End of SUB1 definition
$!
$! Define subroutine SUB2
$!
$ SUB2: SUBROUTINE
    .
    .
    .
$ EXIT
$ ENDSUBROUTINE !End of SUB2 definition
$!
$! Start of main routine. At this point, both SUB1 and SUB2
$! have been defined but none of the previous commands have
$! been executed.
$!
$ START:
$ CALL/OUTPUT=NAMES.LOG SUB1 "THIS IS P1"
    .
    .
    .
$ CALL SUB2 "THIS IS P1" "THIS IS P2"
    .
    .
    .
$ EXIT  !Exit this command procedure file
```

The command procedure in this example shows how to use CALL to transfer control to labeled subroutines. The example also shows that you can call a subroutine or another command file from within a subroutine. The CALL command invokes the subroutine SUB1, directing output to the file NAMES.LOG and allowing other users write access to the file. The subroutine SUB2 is called from within SUB1. The procedure executes SUB2 and then uses the @ (Execute Procedure) command to invoke the command procedure FILE.COM. When all the commands in SUB1 have executed,

the CALL command in the main procedure calls SUB2 a second time. The procedure continues until SUB2 has executed.

CANCEL

Cancels wakeup requests for a specified process, including wakeups scheduled with either the RUN command or the $SCHDWK system service.

Requires one of the following:

- **Ownership of the process.**
- **GROUP privilege to cancel scheduled wakeups for processes in the same group but not owned by you.**
- **WORLD privilege to cancel scheduled wakeups for any process in the system.**

format

CANCEL *[process-name]*

parameter

process-name
Specifies a string of 1 to 15 alphanumeric characters. Specifies the name of the process for which wakeup requests are to be canceled. The specified process must have the same group number in its user identification code (UIC) as the current process. If both the /IDENTIFICATION qualifier and the process name are specified, the process name is ignored. If neither the process-name parameter nor the /IDENTIFICATION qualifier are specified, the CANCEL command cancels scheduled wakeup requests for the current (that is, the issuing) process.

qualifier

/IDENTIFICATION=pid
Identifies the process by its process identification (PID). You can omit leading zeros when you specify the PID.

example

```
$ RUN/SCHEDULE=14:00 STATUS
%RUN-S-PROC_ID, identification of created process is 0013012A
    .
    .
    .
$ CANCEL/IDENTIFICATION=13012A
```

The RUN command in this example creates a process to execute the image STATUS. The process hibernates and is scheduled to be awakened at 14:00. Before the process is awakened, the CANCEL command cancels the wake-up request.

CLOSE

Closes a file opened with the OPEN command and deassigns the associated logical name.

format

CLOSE *logical-name[:]*

parameter

logical-name[:]
Specifies the logical name assigned to the file when it was opened with the OPEN command.

qualifiers

/ERROR=label
Specifies a label in the command procedure to receive control if the CLOSE operation results in an error. Overrides any ON condition action specified. If an error occurs and the target label is successfully given control, the global symbol $STATUS retains the code for the error that caused the error path to be taken.

/LOG (default)
/NOLOG
Generates a warning message when you attempt to close a file that was not opened by DCL. If you specify the /ERROR qualifier, the /LOG qualifier has no effect. If the file has not been opened by DCL, the error branch is taken and no message is displayed.

example

```
$ OPEN/READ INPUT_FILE   TEST.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=NO_MORE  INPUT_FILE  DATA_LINE
   .
   .
   .
$ GOTO READ_LOOP
$ NO_MORE:
$ CLOSE INPUT_FILE
```

The OPEN command in this example opens the file TEST.DAT and assigns it the logical name of INPUT_FILE. The /END_OF_FILE qualifier on the READ command requests that, when the end-of-file is reached, the command interpreter should transfer control to the line at the label NO_MORE. The CLOSE command closes the input file.

CONNECT

Connects your physical terminal to a virtual terminal that is connected to another process.

You must connect to a virtual terminal that is connected to a process with your user identification code (UIC). No other physical terminals may be connected to the virtual terminal.

format

CONNECT *virtual-terminal-name*

parameter

virtual-terminal-name
Specifies the name of the virtual terminal to which you are connecting. A virtual terminal name always begins with VTA. To determine the name of the virtual terminal that is connected to a process, enter the SHOW USERS command.

qualifiers

/CONTINUE
/NOCONTINUE (default)
Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This allows an interrupted image to continue processing after you connect to another process. The /CONTINUE qualifier is incompatible with the /LOGOUT qualifier.

/LOGOUT (default)
/NOLOGOUT

Logs out your current process when you connect to another process using a virtual terminal. The /LOGOUT qualifier is incompatible with the /CONTINUE qualifier.

example

```
$ RUN AVERAGE
CTRL/Y
$ CONNECT/CONTINUE VTA72
```

In this example, the RUN command is used to execute the image AVERAGE.EXE. This command is entered from a terminal that is connected to a virtual terminal. Next, CTRL/Y is entered to interrupt the image. After you interrupt the image, enter the CONNECT command with the /CONTINUE qualifier. This issues the CONTINUE command, so the image continues to run and connects you to another virtual terminal. You can reconnect to the process later.

CONTINUE

Resumes execution of a DCL command, a program, or a command procedure that was interrupted by CTRL/Y or CTRL/C. You cannot resume execution of the image if you have entered a command that executes another image or if you have invoked a command procedure. You can abbreviate the CONTINUE command to a single letter, C.

format

CONTINUE

parameters

None.

example

```
$ RUN  MYPROGRAM_A
CTRL/Y
$ SHOW TIME
  15-APR-1988 13:40:12
$ CONTINUE
```

In this example, the RUN command executes the program MYPROGRAM_A. While the program is running, pressing CTRL/Y interrupts the image. The SHOW TIME command requests a display of the current date and time. The CONTINUE command resumes the image.

CONVERT

Invokes the Convert Utility (CONVERT) to copy records from one file to another, changing the organization and format of the input file to those of the output file.

format

CONVERT *input-file-spec[,...] output-file-spec*

CONVERT/RECLAIM

Invokes the Convert/Reclaim Utility (CONVERT/RECLAIM) to make empty buckets in Prolog 3 indexed files available so that new records can be written in them. If all the records in a bucket have been deleted, that bucket is locked until CONVERT/RECLAIM makes it available. Unlike CONVERT, CONVERT/RECLAIM maintains record file addresses (RFAs). The /RECLAIM qualifier is required.

format

CONVERT/RECLAIM *file-spec*

COPY

Creates a new file from one or more existing files. If device or directory is not specified, your current default device and directory are used.

format

COPY *input-file-spec[,...] output-file-spec*

parameters

input-file-spec[,...]
Specifies the name of an existing file to be copied. Wildcard characters are allowed. Use a plus sign (+) or a comma (,) to indicate multiple file specifications.

output-file-spec
Specifies the name of the output file into which the input is copied. You must specify at least one field in the output file specification. If the device or directory is not specified, your current default device and directory are used. The asterisk wildcard character can be used in place of any two of the following: the file name, file type, or version number.

qualifiers

/ALLOCATION=n

Output-file-spec qualifier. Forces the initial allocation of the output file to the number of 512-byte blocks specified by n. If not specified, the initial allocation of the output file is determined by the size of the input file being copied.

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONCATENATE (default)
/NOCONCATENATE

Creates one output file from multiple input files when wildcard characters are not used in the output file specification. A specification of /NOCONCATENATE generates multiple output files. Files from Files–11 Structure Level 2 disks are concatenated in alphanumeric order; if you specify a wildcard in the file version field, files are copied in descending order by version number. Files from Files–11 Structure Level 1 disks are concatenated in random order.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each COPY operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CONTIGUOUS
/NOCONTIGUOUS

Output-file-spec qualifier. Specifies that the output file must occupy contiguous physical disk blocks. By default, the COPY command creates an output file in the same format as the corresponding input file and does not report an error if not enough space exists for a contiguous allocation.

The /CONTIGUOUS qualifier has no effect when you copy files to or from tapes because the size of the file on tape cannot be determined until after it is copied to the disk. If you copy a file from a tape and want the file to be contiguous, use the COPY command twice: once to copy the file from the tape, and a second time to create a contiguous file.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the COPY operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or /SINCE qualifiers. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXTENSION=n
Output-file-spec qualifier. Specifies the number of blocks to be added to the output file each time the file is extended.

/LOG
/NOLOG (default)
Controls whether the COPY command displays the file specifications of each file copied.

When you use the /LOG qualifier, the COPY command displays the following for each copy operation: (1) the file specifications of the input and output files, (2) the number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis), and (3) the total number of new files created.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. The /MODIFIED qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/OVERLAY
/NOOVERLAY (default)
Output-file-spec qualifier. Requests that data in the input file be copied into the existing specified file, overlaying the existing data, rather than allocating new space for the file. The physical location of the file on disk does not change.

/PROTECTION=(code)
Output-file-spec qualifier. Specifies protection for the output file. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (read), W (write), E (execute), or D (delete). The default protection is that of the existing output file. If no output file exists, the current default protection applies.

/READ_CHECK
/NOREAD_CHECK (default)
Input-file-spec qualifier. Reads each record in the input files twice to verify that it has been read correctly.

/REPLACE
/NOREPLACE (default)

Output-file-spec qualifier. Requests that, if a file already exists with the same file specification as that entered for the output file, the existing file is to be deleted. The COPY command allocates new space for the output file. In general, when you use the /REPLACE qualifier, you will want to include version numbers with the file specifications. By default, the COPY command creates a new version of a file if a file with that specification already exists, incrementing the version number. With /NOREPLACE, an error is signaled when a conflict in version numbers occurs.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/TRUNCATE
/NOTRUNCATE (default)

Output-file-spec qualifier. Controls whether or not the COPY command truncates an output file at the end-of-file when copying it. By default, the size of the output file is determined by the allocation of the input file.

/VOLUME=n

Output-file-spec qualifier. Places the output file on the specified relative volume number of a multivolume set. By default, the output file is placed arbitrarily in a multivolume set.

/WRITE_CHECK
/NOWRITE_CHECK (default)

Output-file-spec qualifier. Reads each record in the output file after it was written to verify that the record was successfully copied and that the file can subsequently be read without error.

example

```
$ COPY/LOG  A.DAT,B.MEM C.*
%COPY-S-COPIED, DBA0:[MAL]A.DAT;5 copied to DBA0:[MAL]C.DAT;11 (1 block)
%COPY-S-COPIED, DBA0:[MAL]B.MEM;2 copied to DBA0:[MAL]C.MEM;24 (58 records)
%COPY-S-NEWFILES, 2 files created
```

In this example, the two input file specifications are separated with a comma. The asterisk wildcard character in the output file specification indicates that two output files are to be created. For each copy operation, the COPY command uses the file type of the input file to name the output file.

CREATE

Creates a sequential text file (or files). Specify the content of the file on the lines following the command, one record per line. In interactive mode, terminate the file input with CTRL/Z. In a command procedure, terminate the file input with a line beginning with a dollar sign in column 1 (or with the end of the command procedure).

format

CREATE *file-spec[,...]*

parameter

file-spec[,...]
Specifies the name of one or more input files to be created. Wildcard characters are not allowed. If you omit either the file name or the file type, the CREATE command does not supply any defaults. The file name or file type is null. If the specified file already exists, a new version is created.

qualifiers

/LOG
/NOLOG (default)
Displays the file specification of each new file created as the command executes.

/OWNER_UIC=uic
Requires SYSPRV privilege to specify a UIC other than your own.
Specifies the user identification code (UIC) to be associated with the file being created.

/PROTECTION=(code)
Specifies protection for the file. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and protection as R (read), W (write), E (execute), or D (delete). If you do not specify a value for each access category, the command applies the current default protection for each unspecified category.

/VOLUME=n
Places the file on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily in a multivolume set.

example

```
$ CREATE MEET.TXT
John, Residents in the apartment complex will hold their annual meeting
this evening.  We hope to see you there, Regards, Elwood
CTRL/Z
```

The CREATE command in this example creates a text file named MEET.TXT in your default directory. The text file MEET.TXT contains the lines that follow until the CTRL/Z.

CREATE/DIRECTORY

Creates one or more new directories or subdirectories. The /DIRECTORY qualifier is required.

Requires WRITE (W) access to the master file directory (MFD) to create a first-level directory. Requires WRITE access to the lowest level directory that currently exists to create a subdirectory.

format

CREATE/DIRECTORY *directory-spec[,...]*

parameter

directory-spec[,...]
Specifies the name of one or more directories or subdirectories to be created. The directory specification optionally can be preceded by a device name (and colon). The default is the current default directory. Wildcard characters are not allowed. When creating a subdirectory, separate the names of the directory levels with periods.

qualifiers

/LOG
/NOLOG (default)
Controls whether the CREATE/DIRECTORY command displays the directory specification of each directory after creating it.

/OWNER_UIC[=option]
Requires SYSPRV privilege for a UIC (user identification code) other than your own.

Specifies an owner UIC for the directory. The default is your UIC. You can specify the keyword PARENT in place of a UIC to mean the UIC of the parent (next-higher-level) directory. If a user with privileges creates a subdirectory, by default, the owner of the subdirectory will be the owner of the parent directory (or the owner of the Master File Directory, if creating a main level directory). If you do not specifiy the /OWNER_UIC qualifier when creating a directory, the command assigns ownership as follows: (1) if you specify the directory name in either alphanumeric or subdirectory format, the default is your UIC (unless you are privileged in which case the UIC defaults to the parent directory); (2) if you specify the directory in UIC format, the default is the specified UIC.

/PROTECTION=(code)
Specifies protection for the directory. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and protection as R (read), W (write), E (execute), or D (delete). The default protection is the protection of the parent directory (the next-higher level directory, or the master directory for top-level directories) minus any delete access.

/VERSION_LIMIT=n

Specifies the number of versions of any one file that can exist in the directory. If you exceed the limit, the system deletes the lowest numbered version. A specification of 0 means no limit. The maximum number of versions allowed is 32,767. The default is the limit for the parent (next-higher-level) directory.

/VOLUME=n

Requests that the directory file be placed on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily within the multivolume set.

example

```
$ CREATE/DIRECTORY/VERSION_LIMIT=2 $DISK1:[ACCOUNTS.MEMOS]
```

In this example, the CREATE/DIRECTORY command creates a subdirectory named MEMOS in the ACCOUNTS directory on $DISK1. No more than two versions of each file can exist in the directory.

CREATE/FDL

Invokes the Create/FDL Utility (CREATE/FDL) to use the specifications in an FDL file to create a new, empty data file. Use this utility to create a data file from a particular FDL specification. The /FDL qualifier is required.

format

CREATE/FDL *=fdl-file-spec [file-spec]*

CREATE/NAME_TABLE

Creates a new logical name table.

format

CREATE/NAME_TABLE *table-name*

parameter

table-name

Specifies a string of 1 to 31 characters that identifies the logical name table you are creating. The string can include alphanumeric characters, the dollar sign, and the underscore.

qualifiers

/ATTRIBUTES[=(keyword[,...])]

Specifies attributes for the logical name table. If you specify only one keyword, you can omit the parentheses. If you do not specify the /ATTRIBUTES qualifier, no attributes are set.

You can specify the following keywords for attributes:

CONFINE	Does not copy the table name or the logical names contained in the table into a spawned subprocess; used only when creating a private logical name table.
NO_ALIAS	No identical names (either logical names or names of logical name tables) may be created in an outer (less privileged) mode in the current directory. Deletes any previously created identical table names in an outer access mode in the same logical name table directory.
SUPERSEDE	Creates a new table that supersedes any previous (existing) table that contains the name, access mode, and directory table that you specify.

/EXECUTIVE_MODE

Requires SYSNAM privilege. Creates an executive mode logical name table.

/LOG (default)
/NOLOG

Controls whether or not an informational message is generated when the SUPERSEDE attribute is specified, or when the table already exists but the SUPERSEDE attribute is not specified. The default is /LOG; that is, the informational message is displayed.

/PARENT_TABLE=table

Requires EXECUTE (E) access to the parent table and SYSPRV privilege to create a shareable logical name table. Specifies the name of the parent table. If you do not specify a parent table, the default table is LNM$PROCESS_DIRECTORY. A shareable table has LNM$SYSTEM_DIRECTORY as its parent table. The parent table must have the same access mode or a higher-level access mode than the one you are creating.

/PROTECTION

Applies the specified protection to shareable name tables. The ownership categories are SYSTEM, OWNER, GROUP, WORLD; the access categories are R (READ), W (WRITE), E (EXECUTE) and D (DELETE). The default protection is (SYSTEM:RWED,OWNER:RWED,GROUP:,WORLD:)

/QUOTA=number-of-bytes

Specifies the size limit of the logical name table. If you do not specify the /QUOTA qualifier, or if you specify /QUOTA=0, the table has unlimited quota.

/SUPERVISOR_MODE (default)
Creates a supervisor mode logical name table. If you do not specify a mode, a supervisor mode logical name table is created.

/USER_MODE
Creates a user mode logical name table. If you do not explicitly specify a mode, a supervisor mode logical name table is created.

example

```
$ CREATE/NAME_TABLE TEST_TAB
$ SHOW LOGICAL TEST_TAB
%SHOW-S-NOTRAN, no translation for logical name TEST_TAB
$ SHOW LOGICAL/TABLE=LNM$PROCESS_DIRECTORY TEST_TAB
```

In this example, the CREATE/NAME_TABLE command creates a new table called TEST_TAB. By default, the name of the table is entered in the process directory. The first SHOW LOGICAL command does not find the name TEST_TAB because it does not, by default, search the process directory table. You must use the /TABLE qualifier to request that the process directory be searched.

DEALLOCATE

Makes an allocated device available to other processes (but does not deassign any logical name associated with the device).

format

DEALLOCATE *device-name[:]*

parameter

device-name[:]
Name of the device to be deallocated. The device name can be a physical device name or a logical name. On a physical device name, the controller defaults to A and the unit to 0. Incompatible with the /ALL qualifier.

qualifier

/ALL
Deallocates all devices currently allocated by your process. Incompatible with the device-name parameter.

example

```
  ALLOCATE   MT:   TAPE
%DCL-I-ALLOC, _MTB1:  allocated
    .
    .
    .
$ DEALLOCATE  TAPE:
```

In this example, the ALLOCATE command requests that any magnetic tape drive be allocated and assigns the logical name TAPE to the device. The response to the ALLOCATE command indicates the successful allocation of the device MTB1. The DEALLOCATE command specifies the logical name TAPE to release the tape drive.

DEASSIGN

Cancels logical name assignments made with the ALLOCATE, ASSIGN, DEFINE, or MOUNT command. The DEASSIGN command also deletes logical name tables created with the CREATE/NAME_TABLE command. Logical names in private tables are deleted automatically when your process terminates. All logical names in the job table and the job table itself are deleted when your process terminates. User mode logical names in the process table are deleted automatically when the next image exits. All other logical names in shareable tables remain unless explicitly deassigned. All names in descendant tables are deleted when the parent table logical name is deassigned.

format

DEASSIGN *[logical-name[:]]*

parameter

logical-name[:]

Specifies the logical name to be deassigned. Logical names can have from 1 to 255 characters. If the logical name contains any characters other than alphanumerics, dollar signs, or underscores, enclose it in quotation marks. The logical-name parameter is required unless you use the /ALL qualifier. If a colon is present in the logical name, you must type two colons in the logical-name parameter of the DEASSIGN command (for example, DEASSIGN FILE::).

qualifiers

/ALL

Deletes all logical names in the same or an outer (less privileged) access mode. If no logical name table is specified, the default is the process table, LNM$PROCESS. If you specify /ALL, you cannot enter a logical-name parameter.

/EXECUTIVE_MODE

Requires SYSNAM privilege to deassign executive mode logical names. Deletes only entries that were created in the specified mode or an outer (less privileged) mode. If you do not have SYSPRV privilege for executive mode, a supervisor mode operation is assumed.

/GROUP

Requires GRPNAM or SYSPRV privilege to delete entries from the group logical name table. Indicates that the specified logical name is in the group logical name table. The /GROUP qualifier is synonymous with /TABLE=LNM$GROUP.

/JOB

Indicates that the specified logical name is in the jobwide logical name table. The /JOB qualifier is synonymous with /TABLE=LNM$JOB. If you do not explicitly specify a logical name table, the default is /PROCESS.

/PROCESS (default)

Indicates that the specified logical name is in the process logical name table. The /PROCESS qualifier is synonymous with /TABLE=LNM$PROCESS.

/SUPERVISOR_MODE (default)

Deletes entries in the specified logical name table that were created in supervisor mode. If you specify the /SUPERVISOR_MODE qualifier, the DEASSIGN command also deassigns user mode entries with the same name.

/SYSTEM

Requires SYSNAM or SYSPRV privilege to delete entries from the system logical name table. Indicates that the specified logical name is in the system logical name table. The /SYSTEM qualifier is synonymous with /TABLE=LNM$SYSTEM.

/TABLE=name

Requires WRITE (W) access to the table to delete a shareable logical name. Requires SYSPRV or DELETE (D) access to delete a shareable logical name table. Specifies the table from which the logical name is to be deleted. Defaults to LNM$PROCESS. The table can be the process, group, job, or system table, one of the directory tables, or the name of a user-created table.

/USER_MODE

Deletes entries in the process logical name table that were created in user mode. If you specify the /USER_MODE qualifier, the DEASSIGN command can deassign only user mode entries.

example

```
$ DEASSIGN/TABLE=LNM$PROCESS_DIRECTORY TAX
```

The DEASSIGN command in this example deletes the logical name table TAX, and any descendant tables. When you delete a logical name table, you must specify either /TABLE=LNM$PROCESS_DIRECTORY or /TABLE=LNM$SYSTEM_DIRECTORY, because the names of all tables are contained in these directories.

DEASSIGN/QUEUE

Deassigns a logical queue from a printer or terminal queue and stops the logical queue. The DEASSIGN/QUEUE command is the complement of the ASSIGN/QUEUE command.

Requires OPER privilege or EXECUTE access to the queue. Cannot be used with batch queues.

format

DEASSIGN/QUEUE *logical-queue-name[:]*

parameter

logical-queue-name[:]

Specifies the name of the logical queue that you want to deassign from a specific printer or terminal queue.

example

```
$ ASSIGN/QUEUE LPA0  ASTER
  .
  .
  .
$ DEASSIGN/QUEUE  ASTER
$ ASSIGN/MERGE LPB0  ASTER
```

The ASSIGN/QUEUE command in this example associates the logical queue ASTER with the print queue LPA0. Later, you deassign the logical queue with the DEASSIGN/QUEUE command. The ASSIGN/MERGE command reassigns the jobs from ASTER to the print queue LPB0.

DEBUG

Invokes the VMS Debugger after program execution is interrupted by CTRL/Y, but only if the /NOTRACEBACK qualifier was not specified with the LINK command when the program was linked.

format

DEBUG

DECK

Marks the beginning of an input stream for a command or program. The DECK command is required in command procedures when the first nonblank character in any data record in the stream is a dollar sign.

Can be used only after a request to execute a command or program that requires input data.

format

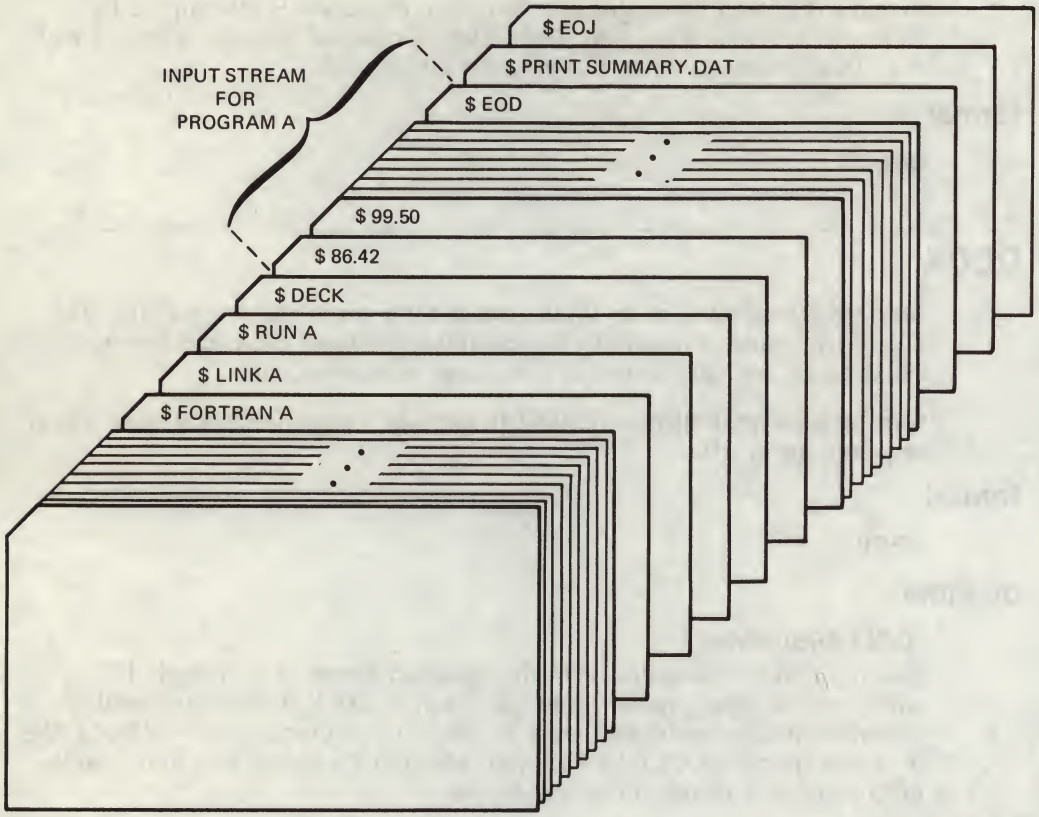
DECK

qualifier

/DOLLARS[=string]

Sets the end-of-file indicator to the specified string of 1 through 15 characters. Enclose the string in quotation marks if it contains literal lowercase letters, multiple blanks, or tabs. If you do not specify /DOLLARS, or if you specify /DOLLARS without specifying a string, you must use the EOD command to signal the end-of-file.

example



ZK-783-82

In this example, the FORTRAN and LINK commands compile and link program A. When the program is run, any data the program reads from the logical device SYS$INPUT is read from the command stream. The DECK command indicates that the input stream can contain dollar signs in column 1 of the record. The EOD command signals end-of-file for the data.

DEFINE

Associates equivalence names with a logical name. If you specify an existing logical name, the new equivalence names replace the existing equivalence name.

format

DEFINE *logical-name equivalence-name[,...]*

parameters

logical-name
Specifies the logical name string, which is a character string containing from 1 to 255 characters. If the logical name is to be entered into the process or system directory logical name tables (LNM$PROCESS_DIRECTORY, LNM$SYSTEM_DIRECTORY), then the name may only have from 1 to 31 alphanumeric characters (including the dollar sign and underscore). If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks (").
Use two consecutive quotation marks ("") to denote an actual quotation mark.

equivalence-name[,...]
Specifies a character string containing from 1 to 255 characters. If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks. Use two consecutive quotation marks ("") to denote an actual quotation mark. Specifying more than one equivalence name for a logical name creates a search list.

qualifiers

/EXECUTIVE_MODE
Requires SYSNAM privilege to create an executive mode logical name. Creates an executive mode logical name in the specified table.

If you specify the /EXECUTIVE_MODE qualifier and you do not have SYSNAM, the DEFINE command ignores the qualifier and creates a supervisor mode logical name. The mode of the logical name must be the same or less privileged than the mode of the table in which you are placing the name.

/GROUP
Requires GRPNAM or SYSPRV privilege to place a name in the group logical name table. Places the logical name in the group logical name table. The /GROUP qualifier is synonymous with /TABLE=LNM$GROUP.

/JOB

Places the logical name in the jobwide logical name table. The /JOB qualifier is synonymous with /TABLE=LNM$JOB.

/LOG (default)
/NOLOG

Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[=(keyword[,...])]

Specifies attributes for a logical name. By default, no attributes are set. Possible keywords are as follows:

CONFINE	The logical name is not copied into a spawned subprocess. This qualifier is relevant only for logical names in a private table.
NO_ALIAS	A logical name cannot be duplicated in the specified table in a less privileged access mode; any previously created identical names in an outer (less privileged) access mode within the specified table are deleted.

If you specify only one keyword, you can omit the parentheses. Only the attributes you specify are set.

/PROCESS (default)

Places the logical name in the process logical name table. The /PROCESS qualifier is synonymous with /TABLE=LNM$PROCESS.

/SUPERVISOR_MODE (default)

Creates a supervisor mode logical name in the specified table. The mode of the logical name must be the same as or less privileged than the mode of the table in which you are placing the name.

/SYSTEM

Requires SYSNAM or SYSPRV privilege to place a name in the system logical name table. Places the logical name in the system logical name table. The /SYSTEM qualifier is synonymous with /TABLE=LNM$SYSTEM.

/TABLE=name

Requires WRITE (W) access to the table to specify the name of a shareable logical name table. Specifies the name of the logical name table in which the logical name is to be entered. You can use the /TABLE qualifier to specify a user-defined logical name table (created with the CREATE/NAME_TABLE command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables.

If you specify the table name using a logical name that has more than one translation, the logical name is placed in the first table found. The default is /TABLE=LNM$PROCESS (or /PROCESS).

/TRANSLATION_ATTRIBUTES[=(keyword[,...])]

Equivalence-name qualifier. Specifies one or more attributes that modify an equivalence string of the logical name. Possible keywords are as follows:

CONCEALED	Indicates that the equivalence string is the name of a concealed device.
TERMINAL	Logical name translation should terminate with the current equivalence string; indicates that the equivalence string should not be translated iteratively.

/USER_MODE

Creates a user mode logical name in the specified table. User mode logical names created within the process logical name tables are used for the execution of a single image.

example

```
$ DEFINE/USER_MODE TM1 $DISK1:[ACCOUNTS.MEMOS]WATER.TXT
```

In this example, the DEFINE command defines TM1 as equivalent to a file specification. After the next image runs, the logical name TM1 is automatically deassigned.

DEFINE/CHARACTERISTIC

Assigns a numeric value to a queue characteristic. The characteristic is created if it does not exist. If a value is already assigned to the characteristic, DEFINE/CHARACTERISTIC alters the assignment of that existing characteristic. The /CHARACTERISTIC qualifier is required. Used in conjunction with the /CHARACTERISTIC qualifier of the PRINT command.

Requires OPER privilege.

format

DEFINE/CHARACTERISTIC *characteristic-name characteristic-number*

parameters

characteristic-name

Assigns a name to the characteristic being defined, which can be the name of an existing characteristic or a string of 1 to 31 characters that defines a new characteristic. The character string can include any uppercase and lowercase letters, digits, the dollar sign ($), and the underscore (_), and must include at least one alphabetic character.

characteristic-number

Assigns a number in the range 0 through 127 to the characteristic being defined.

example

```
$ DEFINE/CHARACTERISTIC REDINK 3
```

The DEFINE command in this example defines the characteristic REDINK with the number 3. When a user enters the command PRINT /CHARACTERISTICS=REDINK (or PRINT /CHARACTERISTICS=3), the job is printed only if the printer queue has been established with the REDINK or 3 characteristic.

DEFINE/FORM

Assigns a numeric value to a print form name and defines the type of physical paper stock. If a value is already assigned to the form name, DEFINE/FORM alters the definition of the existing form. The /FORM qualifier is required. Used in conjunction with the /FORM qualifier of the PRINT command.

Requires OPER privilege.

format

DEFINE/FORM *form-name form-number*

parameters

form-name

Assigns a name to the form being defined. The form name can be the name of an existing form type or a string of 1 to 31 characters that defines a new form type. The character string can include any uppercase and lowercase letters, digits, the dollar sign ($), and the underscore (_), and must include at least one alphabetic character.

form-number

Assigns a number in the range 0 through 999 to the form being defined. The DEFAULT form, which is automatically defined when the system is bootstrapped, is assigned number 0.

qualifiers

/DESCRIPTION=string

A string of up to 255 characters used to describe the form more specifically. The default string is the specified form name. If the string contains alphanumeric, underscore, or dollar sign characters, it must be enclosed in quotation marks (").

/LENGTH=n

Specifies the physical length of a form page in lines. The default page length is 66 lines. The n parameter must be a positive integer greater than 0 and not more than 255.

/MARGIN=(option[,...])

Specifies one or more of the four margin options: BOTTOM, LEFT, RIGHT, and TOP.

BOTTOM=n	Specifies the number of blank lines between the end of the print image area and the end of the physical page; the value of n must be between 0 and the value of the /LENGTH parameter. The default value is 6, which generally means a one-inch bottom margin.
LEFT=n	Specifies the number of blank columns between the leftmost printing position and the print image area; the value of n must be between 0 and the value of the /WIDTH parameter. The default is 0.
RIGHT=n	Specifies the number of blank columns between the /WIDTH parameter and the image area; the value of n must be between 0 and the value of the /WIDTH parameter. The default value is 0.
TOP=n	Specifies the number of blank lines between the top of the physical page and the top of the print image; the value of n must be between 0 and the value of the /LENGTH parameter. The default value is 0.

/PAGE_SETUP=(module[,...])
/NOPAGE_SETUP (default)

Specifies one or more modules that set up the device before every page. The modules are located in the device control library. When a new page is detected, the system extracts the appropriate modules from the device control library and copies them to the printer before the page is printed.

/SETUP=(module[,...])

Specifies one or more modules in the device control library that set up the device appropriately for the specified form. When the form is mounted, the system extracts the specified module from the device control library and copies it to the printer before the file is printed.

/SHEET_FEED
/NOSHEET_FEED (default)

Specifies that print jobs pause at the end of every physical page so that a new sheet of paper can be inserted.

/STOCK=string

Specifies the type of paper stock to be associated with the form. The string parameter can be a string of 1 to 31 characters, including the dollar sign, underscore, and all alphanumeric characters. The default is the form name. If you specify the /STOCK qualifier you must specify the name of the stock to be associated with the form. If you do not specify the /STOCK qualifier, the name of the stock will be the same as the name of the form.

/TRUNCATE (default)
/NOTRUNCATE

Discards any characters that exceed the current line length (specified by /WIDTH and /MARGIN=RIGHT). /TRUNCATE is incompatible with the /WRAP qualifier. If you specify both /NOTRUNCATE and /NOWRAP, the printer prints as many characters on a line as possible.

/WIDTH=n

Specifies the physical width of the paper in terms of columns or character positions. The n parameter must be an integer from 0 through 65,535; the default value is 132. The /MARGIN=RIGHT qualifier overrides the /WIDTH qualifier when determining when to wrap lines of text.

/WRAP
/NOWRAP (default)

Causes lines that exceed the current line length (specified by /WIDTH and /MARGIN=RIGHT) to wrap onto the next line. /WRAP is incompatible with the /TRUNCATE qualifier. If you specify both /NOWRAP and /NOTRUNCATE, the printer prints as many characters on a line as possible.

example

```
$ DEFINE/FORM /MARGIN=(TOP=6,LEFT=10) CENTER 3
```

The DEFINE/FORM command in this example defines the form CENTER to have a top margin of 6 and a left margin of 10. The form is assigned the number 3.

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard. The /KEY qualifier is required.

format

DEFINE/KEY *key-name equivalence-string*

parameters

key-name

Specifies the name of the key that you are defining. The following table lists the key names in column one. The remaining three columns indicate the key designations on the keyboards of the three different types of terminals that allow key definitions.

Key-Name	LK201	VT100-Series	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]
PF4	PF4	PF4	- -
KP0, KP1, ..., KP9	0, 1, ..., 9	0, 1, ..., 9	0, 1, ..., 9
PERIOD	.	.	.

Key-Name	LK201	VT100-Series	VT52
COMMA	,	,	n/a
MINUS	-	-	n/a
ENTER	Enter	ENTER	ENTER
LEFT	←	←	←
RIGHT	→	→	→
Find (E1)	Find	- -	- -
Insert Here (E2)	Insert Here	- -	- -
Remove (E3)	Remove	- -	- -
Select (E4)	Select	- -	- -
Prev Screen (E5)	Prev Screen	- -	- -
Next Screen (E6)	Next Screen	- -	- -
HELP	Help	- -	- -
DO	Do	- -	- -
F6, F7, ..., F20	F6, F7, ..., F20	- -	- -

On LK201 keyboards, you cannot define the UP and DOWN arrow keys or function keys F1 through F5. The LEFT and RIGHT arrow keys and the F6 through F14 keys are reserved for command line editing. You must enter the SET TERMINAL/NOLINE_EDITING command before defining these keys. You can also press CTRL/V to enable keys F7 through F14. Note that CTRL/V will not enable the F6 key.

equivalence-string

Specifies the character string to be processed when you press the key. Enclose the string in quotation marks to preserve spaces and lowercase characters.

qualifiers

/ECHO (default)
/NOECHO

Displays the equivalence string on your screen after the key has been pressed. You cannot use /NOECHO with the /NOTERMINATE qualifier.

/ERASE
/NOERASE (default)

Determines whether the current line is erased before the key translation is inserted.

/IF_STATE=(state-name,...)
/NOIF_STATE

Specifies a list of one or more states, one of which must be in effect for the key definition to work. The /NOIF_STATE has the same meaning as /IF_STATE=current_state.

/LOCK_STATE
/NOLOCK_STATE (default)
Specifies that the state set by the /SET_STATE qualifier remain in effect until explicitly changed. (By default, the /SET_STATE qualifier is in effect only for the next definable key you press or the next read-terminating character that you type.) Can only be specified with the /SET_STATE qualifier.

/LOG (default)
/NOLOG
Displays a message indicating that the key definition has been successfully created.

/SET_STATE=state-name
/NOSET_STATE (default)
Causes the specified state-name to be set when the key is pressed. (By default, the current locked state is reset when the key is pressed.) The state name can be any alphanumeric string; specify the state as a character string enclosed in quotation marks (").

/TERMINATE
/NOTERMINATE (default)
Specifies whether the current equivalence string is to be processed immediately when the key is pressed (equivalent to entering the string and pressing RETURN). By default, you can press other keys before the definition is processed.

example

```
$ DEFINE/KEY PF1 "SHOW " /SET_STATE=GOLD/NOTERMINATE/ECHO
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ DEFINE/KEY PF1 " DEFAULT" /TERMINATE/IF_STATE=GOLD/ECHO
%DCL-I-DEFKEY, GOLD key PF1 has been defined
$ SHOW DEFAULT
DISK1:[JOHN.TEST]
```

In this example, the first DEFINE/KEY command defines the PF1 key to be the string SHOW. The state is set to GOLD for the subsequent key. The /NOTERMINATE qualifier instructs the system not to process the string when the key is pressed. The second DEFINE/KEY command defines the use of the PF1 key when the keypad is in the GOLD state. When the keypad is in the GOLD state, pressing PF1 causes the current read to be terminated.

If you press the PF1 key twice, the system displays and processes the SHOW DEFAULT command.

The word *DEFAULT* in the second line of the example indicates that the PF1 key has been defined in the default state. Note the space before the word *DEFAULT* in the second DEFINE/KEY command. If the space is omitted, the system fails to recognize DEFAULT as the keyword for the SHOW command.

DELETE

Deletes one or more files from a mass storage disk volume.

format

DELETE *file-spec[,...]*

parameter

file-spec[,...]
Specifies the names of one or more files to be deleted from a mass storage disk volume. The first file specification must contain an explicit or default directory specification plus an explicit file name, file type, and version number. Subsequent file specifications need contain only a version number; the defaults will come from the preceding specification. Wildcard characters can be used in any of the file specification fields. If you omit the directory specification or device name, the current default device and directory are assumed. If the file specification contains a null version number (a semicolon followed by no file version number), a version number of 0, or one or more spaces in the version number, the latest version of the file is deleted. To delete more than one file, separate the file specifications with commas or plus signs.

qualifiers

/BACKUP
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]
Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]
Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each DELETE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/ERASE
/NOERASE (default)

When you delete a file, the area in which the file was stored is returned to the system for future use. The data that was stored in that location still exists in the system until new data is written over it. When you specify the /ERASE qualifier, the storage location is overwritten with a system specified pattern so that the data no longer exists.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the DELETE operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and

/MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/LOG
/NOLOG (default)
Controls whether the DELETE command displays the file specification of each file after its deletion.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

example

```
$ DIRECTORY [.SUBTEST]
%DIRECT-W-NOFILES, no files found
$ SET PROTECTION SUBTEST.DIR/PROTECTION=OWNER:D
$ DELETE SUBTEST.DIR;1
```

Before the directory file SUBTEST.DIR is deleted, the DIRECTORY command is used to verify that there are no files cataloged in the directory. The SET PROTECTION command redefines the protection for the directory file so that it can be deleted; then the DELETE command deletes it.

DELETE/CHARACTERISTIC

Deletes the definition of a queue characteristic .

Requires OPER privilege.

format

DELETE/CHARACTERISTIC *characteristic-name*

parameter

characteristic-name
Specifies the name of the characteristic.

example

```
$ DEFINE/CHARACTERISTIC BLUE 7
.
.
.
$ DELETE/CHARACTERISTIC BLUE
$ DEFINE/CHARACTERISTIC BLUE_INK 7
```

The DEFINE/CHARACTERISTIC command in this example establishes the characteristic BLUE, with number 7, to mean blue ink ribbons for printers. To change the name of the characteristic, enter the DELETE/CHARACTERISTIC command. Then enter another DEFINE /CHARACTERISTIC command to rename the characteristic to BLUE_INK, using the characteristic number 7.

DELETE/ENTRY

Deletes one or more print or batch jobs from a queue. The jobs can be in progress or waiting in the queue.

Requires OPER privilege, EXECUTE access to the queue, or DELETE access to the job.

format

DELETE/ENTRY=*(job-number[,...]) [queue-name:]*

parameters

job-number[,...]
Specifies the job number of a job to be deleted from the queue.

queue-name[:]
Specifies the name of the queue where the jobs are located.

example

```
$ PRINT/HOLD   ALPHA.TXT
Job ALPHA (queue SYS$PRINT, entry 110) holding
    .
    .
    .
$ DELETE/ENTRY=110  SYS$PRINT
```

The PRINT command in this example queues a copy of the file ALPHA.TXT in a HOLD status, to defer its printing until a SET QUEUE/ENTRY /RELEASE command is entered. The system displays the job name, entry number, name of the queue in which the job was entered, and the status. Later, the DELETE/ENTRY command requests that the entry be deleted from the queue SYS$PRINT.

DELETE/FORM

Deletes a form type for a printer or a terminal queue previously established with the DEFINE/FORM command. When you delete a form definition, you must ensure that no outstanding references to the form exist in queues that have been mounted with the form or by jobs requesting that form.

Requires OPER privilege.

format

DELETE/FORM *form-name*

parameter

form-name
Specifies the name that was assigned to the form by a DEFINE/FORM command.

example

```
$ DELETE/FORM CENTER
```

The DELETE/FORM command in this example deletes the form named CENTER.

DELETE/INTRUSION_RECORD

Removes an entry from the break-in database.

Requires CMKRNL and SECURITY privileges.

format

DELETE/INTRUSION_RECORD *source*

parameter

source
Source field of the entry to be removed from the break-in database.

example

```
$ DELETE/INTRUSION_RECORD TTC2:
```

In this example, the DELETE/INTRUSION_RECORD command removes all intrusion records generated by break-in attempts on TTC2. No username is specified because none of the login failures occurred for valid users.

DELETE/KEY

Deletes key definitions that have been established by the DEFINE/KEY command.

format

DELETE/KEY *[key-name]*

parameter

key-name
Specifies the name of the key to be deleted. Incompatible with the /ALL qualifier.

qualifiers

/ALL
Deletes all key definitions in the specified state; the default is the current state. If you use the /ALL qualifier, do not specify a key name.

/LOG (default)
/NOLOG
Controls whether messages are displayed indicating that the specified key definitions have been deleted.

/STATE=(state-name[,...])
/NOSTATE (default)

Specifies the name of the state for which the specified key definition is to be deleted. The default state is the current state.

example

```
$ DEFINE/KEY PF3 "SHOW TIME" /TERMINATE
%DCL-I-DEFKEY, DEFAULT key PF3 has been defined
$ PF3
$ SHOW TIME
 15-APR-1988 14:43:59
    .
    .
    .
$ DELETE/KEY PF3
%/DCL-I-DELKEY, DEFAULT key PF3 has been deleted
$ PF3
$
```

In this example, the DEFINE/KEY command defines the PF3 key on the keypad as SHOW TIME. To undefine the PF3 key, use the DELETE/KEY command. When the user presses PF3, only the system prompt is displayed.

DELETE/QUEUE

Deletes a print or batch queue and all the jobs in the queue. The specified queue must be stopped first.

Requires OPER privilege.

format

DELETE/QUEUE *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue to be deleted.

example

```
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START LPA0
    .
    .
    .
$ STOP/QUEUE/NEXT LPA0
$ DELETE/QUEUE LPA0
```

In this example, the first command initializes and starts the printer queue LPA0. The STOP/QUEUE/NEXT command stops the queue. The DELETE/QUEUE command deletes the queue.

DELETE/SYMBOL

Deletes one or all symbol definitions from a local or global symbol table. The /SYMBOL qualifier is required.

format

DELETE/SYMBOL *[symbol-name]*

parameter

symbol-name
Specifies the name of the symbol to be deleted. A name is required unless the /ALL qualifier is specified. The symbol-name parameter is incompatible with the /ALL qualifier.

qualifiers

/ALL
Deletes all symbols from the specified table. The /ALL qualifier is incompatible with the symbol-name parameter.

/GLOBAL
Deletes the symbol from the global symbol table of the current process.

/LOCAL (default)
Deletes the symbol from the local symbol table of the current process.

/LOG
/NOLOG (default)
Controls whether an informational message listing each symbol being deleted is displayed.

example

```
$ DELETE/SYMBOL/LOG FOO
%DCL-I-DELSYM, LOCAL symbol FOO has been deleted
```

In this example, the DELETE/SYMBOL command deletes the symbol FOO from the local symbol table for the current process. In addition, the /LOG qualifier causes an informational message, listing the symbol being deleted, to be displayed.

DEPOSIT

Replaces the contents of the specified locations in virtual memory and displays the new contents. If the specified address can be read but not written by the current access mode, the original contents are displayed; if the specified address can be neither read nor written, asterisks are displayed in the data field. The DEPOSIT command maintains a pointer at that location (at the byte following the last byte modified).

Requires user mode read (R) and write (W) access to the virtual memory location whose contents you wish to change.

format

DEPOSIT *location=data[,...]*

parameters

location

Specifies the starting virtual address or range of virtual addresses (where the second address is larger than the first) whose contents are to be changed. A location can be any valid integer expression containing an integer value, a symbol name, a lexical function, or a combination of these entities. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X). The specified location must be within the virtual address space of the image currently running in the process.

data[,...]

Specifies the data to be deposited into the specified locations. By default, the data is assumed to be in hexadecimal format; it is then converted to binary format and is written into the specified location.

qualifiers

/ASCII

Indicates that the specified data is ASCII. Only one data item is allowed; all characters to the right of the equal sign are considered to be part of a single string. Unless they are enclosed within quotation marks, characters are converted to uppercase and multiple spaces are compressed to a single space before the data is written in memory. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory. When you specify /ASCII, or when ASCII mode is the default, the location you specify is assumed to be hexadecimal.

/BYTE

Requests that data be deposited one byte at a time.

/DECIMAL
Indicates that the data is decimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/HEXADECIMAL
Indicates that the data is hexadecimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/LONGWORD
Requests that data be deposited a longword at a time.

/OCTAL
Indicates that the data is octal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/WORD
Requests that the data be deposited one word at a time.

example

```
$ DEPOSIT/ASCII   2C00=FILE: NAME: TYPE:
00002C00:  FILE: NAME: TYPE:...
```

In this example, the DEPOSIT command deposits character data at hexadecimal location 2C00 and displays the contents of the location after modifying it. Because the current default length is a longword, the response from the DEPOSIT command displays full longwords. Trailing dots (ellipses) indicate that the remainder of the last longword of data contains information that was not modified by the DEPOSIT command.

DIFFERENCES

Compares the contents of two disk files and displays a listing of the records that do not match.

format

DIFFERENCES *input1-file-spec [input2-file-spec]*

parameters

input1-file-spec
Specifies the first file to be compared. The file specification must include a file name and a file type. Wildcard characters are not allowed.

input2-file-spec
Specifies the second file to be compared. Unspecified fields default to the corresponding fields in input1-file-spec. Wildcard characters are not allowed. If you do not specify a secondary input file, the DIFFERENCES command uses the next lower version of the primary input file.

qualifiers

/CHANGE_BAR[=([change-char][,[NO]NUMBER])]

Marks with the specified character in the left margin each line in the input1 file that differs from the corresponding line in the input2 file. If you do not specify a change bar character, the default is an exclamation point (!) for ASCII output. If you specify hexadecimal or octal output (see /MODE qualifier), the change bar character is ignored and differences are marked by a "***CHANGE***" string in the record header. The keyword NONUMBER suppresses line numbers in the listing. If neither the NUMBER nor NONUMBER keyword is specified, the default is controlled by the /[NO]NUMBER command qualifier. If only one option is specified, the parentheses can be omitted.

/COMMENT_DELIMITER[=(character[,...])]

Ignores lines starting with a specified comment character. If the comment character is an exclamation point or semicolon, it can appear anywhere in the line and characters to the right of the character are ignored. If you specify just one character, you can omit the parentheses. Lowercase characters are automatically converted to uppercase unless they are enclosed in quotation marks. Non-alphanumeric characters (such as ! and ,) must be enclosed in quotation marks. You can specify up to 32 comment characters by typing the character itself or one of the following keywords. (Keywords can be abbreviated provided that the resultant keyword is not ambiguous and has at least two characters; single letters are treated as delimiters.)

Keyword	Character
COLON	Colon (:)
COMMA	Comma (,)
EXCLAMATION	Exclamation point (!)
FORM_FEED	Form feed
LEFT	Left bracket ([)
RIGHT	Right bracket (])
SEMI_COLON	Semicolon (;)
SLASH	Slash (/)
SPACE	Space
TAB	Tab

The following characters are the default comment delimiters for files with the specified file types.

File Type	Default Comment Character
B2S, B32, BAS, BLI	!
CBL, CMD	! and ;
COB	* or / in the first column
COM, COR	!
FOR	! anywhere and C, D, c, d in the first column
HLP	!
MAC, MAR	;
R32, REQ	!

/IGNORE=(keyword[,...])

Inhibits the comparison of the specified characters, strings, or records; also controls whether the comparison records are output to the listing file as edited records or exactly as they appeared in the input file. If you specify only one keyword, you can omit the parentheses. The keyword parameter refers either to a character or a keyword. The first set of keywords determines what, if anything, is ignored during file comparison; the second set of keywords determines whether or not ignored characters are included in the output. The following keywords are valid options for the /IGNORE qualifier:

BLANK_LINES	Blank lines between data lines.
COMMENTS	Data following a comment character.
FORM_FEEDS	Form feed character.
HEADER[=n]	First *n* records of the file, beginning with a record whose first character is a form feed. The first record is not ignored if the only character it contains is a form feed. (N indicates the number of records and defaults to 2. A record with a single form feed is not counted.)
TRAILING_SPACES	Space and tab characters at the end of a data line.
SPACING	Extra blank spaces or tabs within data lines.
EDITED	Omits ignored characters from the output records.
EXACT	Includes ignored characters in the output records.
PRETTY	Formats output records.

If you specify /PARALLEL, output records are always formatted. To format output records, specify the following characters:

Character	Formatted Output
Tab (CTRL/I)	1-8 spaces
RETURN (CTRL/M)	<CR>
Line feed (CTRL/J)	<LF>
Vertical tab (CTRL/K)	<VT>
Form feed (CTRL/L)	<FF>
Other nonprinting characters	. (period)

/MATCH=size

Specifies the number of records that should indicate matching data after a difference is found. By default, after DIFFERENCES finds unmatched records, it assumes that the files once again match after it finds three sequential records that match. Use the /MATCH qualifier to override the default match size of 3.

/MAXIMUM_DIFFERENCES=n

Terminates DIFFERENCES after a specified number of unmatched records (specified with the n parameter) is found.

/MERGED[=n]

Specifies that the output file contain a merged list of differences with the specified number of matched records listed after each group of unmatched records. The specified number (the value n) must be less than or equal to the number specified in the /MATCH qualifier. By default, DIFFERENCES produces a merged listing with one matched record listed after each set of unmatched records (that is, /MERGED=1). If neither /MERGED nor /SEPARATED nor /PARALLEL is specified, the resulting output is merged, with one matched record following each unmatched record.

/MODE=(radix[,...])

Specifies the format of the output. You can request that the output be formatted in one or more radix modes by specifying the following keywords, which may be abbreviated: ASCII (default), HEXADECIMAL, or OCTAL. If you specify only one radix, you can omit the parentheses. If you specify /PARALLEL or /SLP, /MODE is ignored for that listing form.

/NUMBER (default)
/NONUMBER

Includes line numbers in the listing of differences.

/OUTPUT[=file-spec]

Specifies an output file to receive the list of differences. By default, the output is written to the current SYS$OUTPUT device. If the file-spec parameter is not specified, the output is directed to the first input file with a file type of DIF. No wildcard characters are allowed.

/PARALLEL[=n]
Lists the records with differences side by side. The value n specifies the number of matched records to merge after each unmatched record; the value n must be a non-negative decimal number less than or equal to the number specified in /MATCH.

/SEPARATED[=(input1-file-spec[,input2-file-spec])]
Lists sequentially only the records from the specified file that contain differences. If no files are specified, a separate listing is generated for each file. If only one file is specified, you can omit the parentheses. To specify the input1-file-spec parameter, use either the first input file specified as the DIFFERENCES parameter or the keyword MASTER. To specify the input2-file-spec parameter, use either the second input file specified as the DIFFERENCES parameter or the keyword REVISION. By default, DIFFERENCES creates only a merged list of differences.

/SLP
Requests that DIFFERENCES produce an output file suitable for input to the SLP editor. If you use the /SLP qualifier, you cannot specify any of the following output file qualifiers: /MERGED, /PARALLEL, /SEPARATED, or /CHANGE_BAR.

Use the output file produced by the SLP qualifier as input to SLP to update the master input file, that is, to make the master input file match the revision input file.

When you specify /SLP and you do not specify /OUTPUT, DIFFERENCES writes the output file to a file with the same file name as the master input file with the file type DIF.

/WIDTH=n
Specifies the width of the lines in the output file. The default is 132 characters. If output is written to the terminal, /WIDTH is ignored and the terminal line width is used.

/WINDOW=size
Searches the number of records specified (the value n) before a record is declared as unmatched. By default, DIFFERENCES searches to the ends of both input files before listing a record as unmatched.

example

```
$ DIFFERENCES EXAMPLE.TXT
************
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
    1   DEMONSTRATION
    2   OF V3.0 DIFFERENCES
    3   UTILITY
******
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
    1   DEMONSTRATION
    2   OF VMS DIFFERENCES
    3   UTILITY
************
Number of difference sections found: 1
Number of difference records found: 2
DIFFERENCES/MERGED=1-
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

In this example, the DIFFERENCES command compares the contents of the two most recent versions of the file EXAMPLE.TXT in the current default directory. DIFFERENCES compares every character in every record and displays the results at the terminal.

DIRECTORY

Provides a list of files or information about a file or group of files.

Requires READ (R) access to the directories or sufficient privilege to override the protection to obtain information. Requires READ access to the files or sufficient privilege to override the protection to obtain information other than the file name.

format

DIRECTORY *[file-spec[,...]]*

parameter

file-spec[,...]

Specifies one or more files to be listed. The syntax of a file specification determines which files will be listed, as follows:

- If you do not enter a file specification, the DIRECTORY command lists all versions of the files in the current default directory.
- If you specify only a device name, the DIRECTORY command uses your default directory specification.
- Whenever the file specification does not include a file name, file type and a version number, all versions of all files in the specified directory are listed.

- If a file specification contains a file name or a file type, or both, and no version number, the DIRECTORY command lists all versions.
- If a file specification contains only a file name, the DIRECTORY command lists all files in the current default directory with that file type, regardless of file type and version number.
- If a file specification contains only a file type, the DIRECTORY command lists all files in the current default directory with that file type, regardless of file name and version number.

Wildcard characters can be used. Separate multiple file specifications with either commas or plus signs.

qualifiers

/ACL
Controls whether the access control list (ACL) is displayed for each file. The /ACL qualifier overrides the /COLUMNS qualifier.

/BACKUP
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]
Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BRIEF (default)
Displays only a file's name, type, and version number. You can use the /ACL, /DATE, /FILE_ID, /NOHEADING, /OWNER, /PROTECTION, /SECURITY, and /SIZE qualifiers to expand a brief display.

/BY_OWNER[=uic]
Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/COLUMNS=n
Specifies the number of columns in a brief display. The default is four. However, you can request as many columns as you like, restricted by the value of the /WIDTH qualifier. The /COLUMNS qualifier is incompatible with /ACL, /FULL, and /SECURITY.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/DATE[=option]
/NODATE (default)

Includes the backup, creation, expiration, or modification date for each specified file; the default is /NODATE. If you use the /DATE qualifier without an option, the creation date is provided. Possible options are as follows:

ALL	Creation, expiration, backup, and last modification dates
BACKUP	Last backup date
CREATED	Creation date
EXPIRED	Expiration date
MODIFIED	Last modification date

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the DIRECTORY operation. When using /EXCLUDE in a DIRECTORY operation of a different device, use only the file name in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/FILE_ID

Controls whether the file's identification number (FID) is displayed. By default, a file's identification is not displayed unless the /FULL qualifier is specified.

/FULL

Displays the following information for each file:

File name
File type
Version number
Number of blocks used

Number of blocks allocated
Date of creation
Date last modified and revision number
Date of expiration
Date of last backup
File owner's UIC
File protection
File identification number (FID)
File organization
Journaling information
Other file attributes
Record attributes
Record format
Access control list (ACL)

/GRAND_TOTAL
Displays only the totals for all files and directories that have been specified.

/HEADING
/NOHEADING
Controls whether heading lines consisting of a device description and directory specification are printed. The default output format provides this heading. When /NOHEADING is specified, the display is in single-column format and the device and directory information appears with each file name. The /NOHEADING qualifier overrides /COLUMNS.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/OUTPUT[=file-spec]
/NOOUTPUT
Controls where the output of the command is sent. By default, the display is written to the current SYS$OUTPUT device. No wildcard characters are allowed.

/OWNER
/NOOWNER (default)
Controls whether the file owner's UIC is listed.

/PRINTER
Puts the display in a file and queues the file to SYS$PRINT for printing under the name given by the /OUTPUT qualifier. If you do not specify the /OUTPUT qualifier, output is directed to a temporary file named DIRECTORY.LIS, which is queued for printing and then deleted.

/PROTECTION
/NOPROTECTION (default)

Controls whether the file protection for each file is listed.

/SECURITY

Controls whether information about file security is displayed; using /SECURITY is equivalent to using the /ACL, /OWNER, and /PROTECTION qualifiers together.

/SELECT=(keyword[,...])

Allows you to select files for display according to size. Choose one of the following keywords:

SIZE=MAXIMUM=n	Displays files that have fewer blocks than the value of n, which defaults to 1,073,741,823. Use with MINIMUM=n to specify a size range for files to be displayed.
SIZE=MINIMUM=n	Displays files that have blocks equal to or greater than the value of n, which defaults to 0. Use with MAXIMUM=n to specify a size range for files to be displayed.
SIZE=(MAXIMUM=n,MINIMUM=m)	Displays files whose blocksize falls within the specified MAXIMUM and MINIMUM range.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/SIZE[=option]
/NOSIZE (default)

Displays the size in blocks of each file. If you omit the option parameter, the default lists the file size in blocks used (USED). Specify one of the following options:

ALL	Lists the file size both in blocks allocated and blocks used
ALLOCATION	Lists the file size in blocks allocated
USED	Lists the file size in blocks used

/TOTAL

Displays only the directory name and total number of files.

/TRAILING
/NOTRAILING

Controls whether trailing lines that provide the following summary information are displayed:

- Number of files listed

- Total number of blocks used per directory
- Total number of blocks allocated
- Total number of directories and total blocks used or allocated in all directories (only if more than one directory is listed)

By default, the output format includes most of this summary information. The /SIZE and /FULL qualifiers determine more precisely what summary information is included. Used by itself, /TRAILING lists the number of files in the directory. Used with /SIZE, /TRAILING lists the number of files and the number of blocks (displayed according to the option of the /SIZE qualifier, FULL or ALLOCATION). Used with /FULL, /TRAILING lists the number of files as well as the number of blocks used and allocated. If more than one directory is listed, the summary includes the total number of directories, the total number of blocks used, and the total number of blocks allocated.

/VERSIONS=n

Specifies the number of versions of a file to be listed. The default is all versions of each file. A value less than 1 is not allowed.

/WIDTH=(keyword[,...])

Formats the width of the display. Possible keywords are as follows:

DISPLAY=n	Specifies the total width of the display as an integer in the range 1 through 256 and defaults to 0 (setting the display width to the terminal width).
FILENAME=n	Specifies the width of the file name field; defaults to 19.
OWNER=n	Specifies the width of the owner field; defaults to 20.
SIZE=n	Specifies the width of the size field; defaults to 6.

example

```
$ DIRECTORY/FULL [JONES.ITALIA]PROJECTIONS.LIS

Directory WORK:[JONES.ITALIA]

PROJECTIONS.LIS;1            File ID:  (7449,36222,2)
Size:            21/21          Owner:    [DOC,JONES]
Created:   5-MAY-1988 15:49:03.11
Revised:   5-MAY-1988 15:49:49.39 (2)
Expires:   <None specified>
Backup:    <No backup recorded>
File organization:  Sequential
File attributes:    Allocation: 21, Extend: 0, Global buffer count: 0,
                    No version limit
Record format:      Variable length, maximum 80 bytes
Record attributes:  Carriage return carriage control
Journaling enabled: None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 21/21 blocks.
```

The DIRECTORY command in this example shows the date/time format using the default language, English, and the default VMS format. Users also may select other languages and formats that have been defined on their systems with Run-Time Library international date/time formatting routines.

DISCONNECT

Breaks the connection between a physical terminal and a virtual terminal. After the physical terminal is disconnected, both the virtual terminal and the process using it remain on the system.

Requires that your physical terminal is connected to a virtual terminal.

format

DISCONNECT

parameters

None.

qualifier

/CONTINUE
/NOCONTINUE (default)
Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This permits an interrupted image to continue processing after the disconnect takes place.

example

```
$ RUN PAYROLL
CTRL/Y
$ DISCONNECT/CONTINUE
```

In this example, the RUN command is issued from a physical terminal that is connected to a virtual terminal. After the image PAYROLL.EXE is interrupted, the DISCONNECT command disconnects the physical and the virtual terminals without logging out the process. The /CONTINUE qualifier allows the image PAYROLL.EXE to continue to execute. However, the terminal can be used to log in again and perform other work.

DISMOUNT

Closes a mounted disk or magnetic tape volume for further processing and deassigns the logical name associated with the device. If the volume is mounted with the /SHARE qualifier, its logical name is deassigned but the volume remains mounted until all processes using it dismount it or terminate. Note that all open files on the volume must be closed before the actual dismount can be done. Note, also, that the file system cannot dismount a volume while any known file lists associated with it contain entries.

Requires the GRPNAM and SYSNAM user privileges to dismount group and system volumes.

format

DISMOUNT *device-name[:]*

parameter

device-name[:]
Name of the device containing the volume—either a logical name or a physical name. If a physical name is specified, the controller defaults to A and the unit defaults to 0.

qualifiers

/ABORT
Requires volume ownership or the user privilege VOLPRO to use this qualifier with a volume that is mounted neither group nor system.
Specifies that the volume is to be dismounted, regardless of who actually mounted it. The primary purpose of the /ABORT qualifier is to terminate mount verification. DISMOUNT/ABORT also cancels any outstanding I/O requests. If the volume was mounted with the /SHARE qualifier, the /ABORT qualifier causes the volume to be dismounted for all of the users who mounted it.

/CLUSTER
Dismounts a volume clusterwide. After the DISMOUNT command successfully dismounts the volume on the local node, the volume is dismounted on every other node in the existing VAXcluster. If the system is not a member of a VAXcluster, the /CLUSTER qualifier has no effect.

/UNIT
Dismounts only the volume of a volume set on the specified device. By default, all volumes in a set are dismounted.

NOTE: Avoid dismounting the root volume of a volume set, because it contains the master file directory (MFD). It may be impossible to access files on a volume set if the MFD is not accessible.

/UNLOAD (default)
/NOUNLOAD

Unloads the device on which the volume is mounted. If you specify /NOUNLOAD, the device remains in a ready state.

example

```
$ MOUNT MT: PAYVOL TAPE
  .
  .
  .
$ DISMOUNT TAPE:
```

The MOUNT command in this example mounts the tape whose volume identification is PAYVOL on the device MTA0: and assigns the logical name TAPE to the device. By default, the volume is not shareable. The DISMOUNT command releases access to the volume, deallocates the device, and deletes the logical name TAPE.

DUMP

Displays the contents of a file, disk volume, or magnetic tape volume in decimal, hexadecimal, or octal format, as well as the ASCII conversion.

format

DUMP *file-spec [,...]*

parameter

file-spec

Specifies the file or name of the device being dumped.

qualifiers

/ALLOCATED

Includes in the dump all blocks allocated to the file. (By default, the dump does not include blocks following the end-of-file.) /ALLOCATED and /RECORDS are mutually exclusive.

/BLOCKS[=(option[,...])]

Dumps the specified blocks one block at a time, which is the default method for all devices except network devices. Block numbers are specified as integers relative to the beginning of the file. Typically, blocks are numbered beginning with 1. If a disk device is mounted /FOREIGN, blocks are numbered beginning with 0. Select a range of blocks to be dumped by specifying one of the following options:

START:n	Specifies the number of the first block to be dumped; the default is the first block.
END:n	Specifies the number of the last block to be dumped; the default is the last block or the end-of-file block, depending on the /ALLOCATED qualifier.
COUNT:n	Specifies the number of files to be dumped. COUNT provides an alternative to END; you may not specify both.

If you specify only one option, you can omit the parentheses. /BLOCKS and /RECORDS are mutually exclusive.

/BYTE
Formats the dump in bytes. /BYTE, /LONGWORD, and /WORD are mutually exclusive. The default format is composed of longwords.

/DECIMAL
Dumps the file in decimal radix. /DECIMAL, /HEXADECIMAL (default), and /OCTAL are mutually exclusive.

/FILE_HEADER
Dumps each data block that is a valid Files–11 header in Files–11 header format rather than the selected radix and length.

/FORMATTED (default)
/NOFORMATTED
Dumps the file header in Files–11 format; /NOFORMATTED dumps the file header in octal format. This qualifier is useful only when /HEADER is specified.

/HEADER
Dumps the file header and access control list. To dump only the file header, and not the file contents, also specify /BLOCK=(COUNT:0). /HEADER is invalid for devices mounted /FOREIGN.

/HEXADECIMAL (default)
Dumps the file in hexadecimal radix. /DECIMAL, /HEXADECIMAL (default), and /OCTAL are mutually exclusive.

/LONGWORD (default)
Formats the dump in longwords. /BYTE, /LONGWORD, and /WORD are mutually exclusive.

/NUMBER[=n]
Specifies how byte offsets are assigned to the lines of output. If you specify /NUMBER, the byte offsets increase continuously through the dump, beginning with n; if you omit /NUMBER, the first byte offset is 0. By default, the byte offset is reset to 0 at the beginning of each block or record.

/OCTAL
Dumps the file in octal radix. /DECIMAL, /HEXADECIMAL (default), and /OCTAL are mutually exclusive.

/OUTPUT[=file-spec]

Specifies the output file for the dump. If you do not specify a file specification, the default is the file name of the file being dumped and the file type DMP. If /OUTPUT is not specified, the dump goes to SYS$OUTPUT. No wildcard characters are allowed. /OUTPUT and /PRINTER are mutually exclusive.

/PRINTER

Queues the dump to SYS$PRINT in a file named with the file name of the file being dumped and the file type DMP. If /PRINTER is not specified, the dump goes to SYS$OUTPUT. No wildcard characters are allowed. /OUTPUT and /PRINTER are mutually exclusive.

/RECORDS[=(option[,...])]

Dumps the file a record at a time rather than a block at a time. (By default, input is dumped one block at a time for all devices except network devices.) Blocks are numbered beginning with 1.

Select a range of blocks to be dumped by specifying one of the following options:

START:n	Specifies the number of the first record to be dumped; the default is the first record.
END:n	Specifies the number of the last record to be dumped; the default is the last record of the file.
COUNT:n	Specifies the number of records to be dumped. COUNT provides an alternative to END; you may not specify both.

If you specify only one option, you can omit the parentheses. If you specify /RECORDS, you cannot specify /ALLOCATED or /BLOCKS.

/WORD

Formats the dump in words. /BYTE, /LONGWORD, and /WORD are mutually exclusive.

example

```
$ DUMP TEST.DAT
Dump of file DISK0:[NORMAN]TEST.DAT;1 on 15-APR-1988 15:43:26.08
File ID (3134,818,2)   End of file block 1 / Allocated 3
Virtual block number 1 (00000001), 512 (0200) bytes
 706D6173 20612073 69207369 68540033 3.This is a samp 000000
 73752065 62206F74 20656C69 6620656C le file to be us 000010
 61786520 504D5544 2061206E 69206465 ed in a DUMP exa 000020
 00000000 00000000 0000002E 656C706D mple........... 000030
 00000000 00000000 00000000 00000000 ................ 000040
 00000000 00000000 00000000 00000000 ................ 000050
 00000000 00000000 00000000 00000000 ................ 000060
          .
          .
          .
 00000000 00000000 00000000 00000000 ................ 0001E0
 00000000 00000000 00000000 00000000 ................ 0001F0
```

The DUMP command displays the contents of TEST.DAT both in hexadecimal longword format and in ASCII beginning with the first block in the file.

EDIT/ACL

Invokes the Access Control List (ACL) Editor to create or modify an access control list for a specified object. The /ACL qualifier is required.

format

EDIT/ACL *object-spec*

EDIT/EDT

Invokes the VAX EDT interactive text editor. The /EDT qualifier is not required, because EDT is the VMS default editor.

format

EDIT *file-spec*

parameter

file-spec

Specifies the file to be created or edited using the EDT editor. If the file does not exist, it is created by EDT. The EDT editor does not provide a default file type when creating files; if you do not include a file type, it is null. The file must be a disk file on a Files–11 formatted volume. No wildcard characters are allowed in the file specification.

qualifiers

/COMMAND[=file-spec]
/NOCOMMAND

Determines whether or not EDT uses a startup command file. The /COMMAND file qualifier should be followed by an equal sign and the specification of the command file. The default file type for command files is EDT. No wildcard characters are allowed in the file specification. If you do not include the /COMMAND=command file qualifier, EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state. To prevent EDT from processing either the systemwide startup command file or the EDTINI.EDT file in your default directory, use the /NOCOMMAND qualifier as follows:

```
$ EDIT/NOCOMMAND MEMO.DAT
```

/CREATE (default)
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found.

/JOURNAL[=journal-file]
/NOJOURNAL

Determines whether EDT keeps a journal file during your editing session. A journal file contains a record of the keystrokes you enter during an editing session. The default file name for the journal file is the same as the input file name. The default file type is JOU. The /JOURNAL qualifier enables you to use a different file specification for the journal file. If you are editing a file from another directory and want the journal file to be located in that directory, you must use the /JOURNAL qualifier with a file specification that includes the directory name. Otherwise, EDT creates the journal file in the default directory. The directory that is to contain the journal file should not be write-protected. Once you have created a journal file, use EDT/RECOVER to execute the commands in the journal file. No wildcard characters are allowed in the file specification.

/OUTPUT=output-file
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. The default file specification for both the input file and the output file is the same. Use the /OUTPUT qualifier to give the output file a different file specification from the input file. The /NOOUTPUT qualifier suppresses the creation of an output file, but not the creation of a journal file. A system interruption does not prevent you from re-creating your editing session because a journal file is still being maintained. To save your editing session,

even when you specify /NOOUTPUT, use the line mode command WRITE to put the text in an external file before you end the session. No wildcard characters are allowed in the file specification.

/READ_ONLY
/NOREAD_ONLY (default)
Determines whether EDT keeps a journal file and creates an output file. With the default /NOREAD_ONLY, EDT maintains the journal file and creates an output file when it processes the line mode command EXIT. Using the /READ_ONLY qualifier has the same effect as specifying both the /NOJOURNAL and /NOOUTPUT qualifiers. Use /READ_ONLY when you are searching a file and do not intend to make any changes to it. To modify the file, use the line mode command WRITE to save your changes. Remember, however, that you have no journal file.

/RECOVER
/NORECOVER (default)
Determines whether or not EDT reads a journal file at the start of the editing session. When you use the /RECOVER qualifier, EDT reads the appropriate journal file and processes whatever commands it contains. If the journal file type is not JOU or the file name is not the same as the input file name, you must include both the /JOURNAL qualifier and the /RECOVER qualifier as follows:

```
$ EDIT/RECOVER/JOURNAL=SAVE.XXX MEMO.DAT
```

example

```
$ EDIT/OUTPUT=NEWFILE.TXT OLDFILE.TXT
    1       This is the first line of the file OLDFILE.TXT.
*
```

This EDIT command invokes the EDT editor to edit the file OLDFILE.TXT. EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state. When the session ends, the edited file has the name NEWFILE.TXT.

EDIT/FDL

Invokes the VMS FDL Editor (EDIT/FDL) to create and modify File Definition Language (FDL) files. The /FDL qualifier is required.

format

EDIT/FDL *file-spec*

EDIT/SUM

Invokes the SUMSLP batch-oriented editor, to update a single input file with multiple files of edit commands.

format

EDIT/SUM *input-file*

EDIT/TECO

Invokes the TECO interactive text editor. The /TECO qualifier is required.

format

EDIT/TECO *[file-spec]*

EDIT/TECO/EXECUTE=command-file *[argument]*

parameter

file-spec

Specifies the file to be created or edited using the TECO editor. If the file does not exist, it is created by TECO, unless you specify /NOCREATE. No wildcard characters are allowed in the file specification.

qualifiers

/COMMAND[=file-spec]
/NOCOMMAND

Controls whether a startup command file is used. The /COMMAND file qualifier may be followed by an equal sign and the specification of the command file. The default file type for command files is TEC. If you do not include the /COMMAND qualifier, or if you enter /COMMAND without specifying a command file, TECO looks for the TEC$INIT logical name assignment. If TEC$INIT is not defined, no startup commands are executed. No wildcards are allowed in the file specification.

/CREATE (default)
/NOCREATE

Creates a new file when the specified input file cannot be found. If /MEMORY is specified and no input file is specified, the file created is the one specified by the logical name TEC$MEMORY. Normally, TECO creates a new file to match the input file specification if it cannot find the requested file name in the specified directory. When you use /NOCREATE in the TECO command line and type a specification for a file that does not exist, TECO displays an error message and returns you to the DCL command level. /EXECUTE is not compatible with /CREATE and /NOCREATE.

/EXECUTE=command-file [argument]

Invokes TECO and executes the TECO macro found in the command file. The argument, if specified, appears in the text buffer when macro execution starts. Blanks or special characters must be enclosed in quotes. /EXECUTE is incompatible with /CREATE and /MEMORY.

/MEMORY (default)
/NOMEMORY

Specifies that the last file you edited with TECO, identified by the logical name TEC$MEMORY, will be the file edited if you omit the file specification to the EDIT/TECO command.

/OUTPUT=output-file
/NOOUTPUT (default)

Controls how the output file is named at the end of your editing session. By default, the output file has the same name as the input file but is given the next higher available version number. Use the /OUTPUT qualifier to give the output file a file specification different from the input file. No wildcard characters are allowed in the file specification.

/READ_ONLY
/NOREAD_ONLY (default)

Controls whether or not an output file is created. By default, an output file is created; /READ ONLY suppresses the creation of the output file.

example

```
$ EDIT/TECO/OUTPUT=NEWFILE.TXT OLDFILE.TXT
```

This EDIT command invokes the TECO editor to edit the file OLDFILE.TXT. TECO looks for the TEC$INIT logical name assignment. If TEC$INIT is not defined, TECO begins the editing session without using a command file. When the session ends, the edited file has the name NEWFILE.TXT.

EDIT/TPU

Invokes the VAX Text Processing Utility (VAXTPU), running the interactive text editor Extensible VAX Editor (EVE). For more information about using EVE, see Chapter 8. For more information about EVE commands, see the Reference Section.

format

EDIT/TPU *[file-spec]*

parameter

file-spec

Optionally, specifies the input file to be created or edited.

- If you specify an input file on the command line, EVE creates an edit buffer with the same name as that file.
- If you do not specify an input file, EVE creates an empty buffer named MAIN. You can then use the EVE command GET FILE to specify the file you want to edit or create; also you can edit the buffer MAIN and, at the completion of your editing session, specify the output file by using the EVE command WRITE FILE.

Processing wildcards in the file specification depends on the VAXTPU application you are using. With EVE, you can use wildcards in the input file specification. For example:

```
$ EDIT/TPU *.TXT
```

If more than one file matches your request, EVE displays a list of files with the same file type and prompts for more information.

qualifiers

/COMMAND[=command-file] (default)
/NOCOMMAND

Determines whether a user-written command file is used. User-written command files contain VAXTPU procedures and statements to extend or modify the editor or to define a special text-processing environment for creating your own application. The default file type is TPU. You cannot use wildcards in the file specification.

- If you use /COMMAND without specifying a file, VAXTPU tries to read a file called TPU$COMMAND.TPU from your current (default) directory.
- To specify a particular command file, define the logical name TPU$COMMAND or use /COMMAND= and specify the file.
- If a command you specify is not found, the editing session is aborted and you are returned to the DCL level.

/CREATE (default)
/NOCREATE

Controls whether VAXTPU creates an editing buffer when the specified input file is not found. Processing this qualifier depends on the VAXTPU application you are using. By default, EVE creates a buffer in which to edit a file. If you write out the contents of the buffer (with the built-in procedure WRITE_FILE or the EVE command WRITE FILE, or by exiting from the editor), VAXTPU creates a new file or new version of an existing file. If you use /NOCREATE and specify an input file that does not exist, the editing session is aborted and you are returned to the DCL command level.

/DEBUG[=debugger-file]
/NODEBUG (default)

Determines whether a debugger is available during an editing session. Using /DEBUG, with or without specifying a debugger file name, makes a debugger available.

If you use the /DEBUG qualifier but do not specify a debugger file name, VAXTPU reads, compiles, and executes the VAXTPU debugger, SYS$SHARE:TPU$DEBUG.TPU. VAXTPU runs this file before it runs either TPU$INIT_PROCEDURE or the file specified with the /COMMAND qualifier.

/DISPLAY[=file-spec] (default)
/NODISPLAY

Determines whether a VAXTPU session is run from a supported terminal and uses terminal functions such as the screen display and the keyboard. By default, the session is run with the screen management file, TPU$CCTSHR.EXE, for terminals that respond to ANSI control functions and that operate in ANSI mode. VAXTPU expects sessions to be run from a supported terminal unless you specify /NODISPLAY. The /NODISPLAY qualifier causes VAXTPU to run without using the screen display and the keyboard functions of a terminal. Typically, you use /NODISPLAY when running VAXTPU procedures in a batch job or when using VAXTPU on an unsupported terminal.

If you use /NODISPLAY, window and screen manipulation commands and key definitions cause errors. When using /NODISPLAY, you must use a special startup file (either a section file or a command file) for the session. This startup file should follow these rules:

- The file should not include statements for key definitions or screen manipulation, except for READ_LINE, MESSAGE, and LAST_KEY, which work with some restrictions.
- The file should be a complete VAXTPU session; it should end with EXIT or QUIT.

/INITIALIZATION[=initialization-file]
/NOINITIALIZATION

Determines whether VAXTPU executes a user-written initialization file containing editor commands, typically to set editing values or private defaults (such as margins and tab stops). The default for this qualifier depends on the section file used during the editing session. Initialization files differ from command files in that initialization files contain EVE commands, while command files contain VAXTPU procedures and statements. The default file type for initialization files is EVE.

With EVE, if you use /INITIALIZATION without specifying an initialization file, VAXTPU searches your current (default) directory and SYS$LOGIN for a file called EVE$INIT.EVE. To specify a particular initialization file, define the logical name EVE$INIT or use /INITIALIZATION= and specify the file you want. You cannot use wildcards in the file specification, and you cannot nest initialization files.

If you do not want to use any initialization file, use /NOINITIALIZATION.

/JOURNAL[=journal-file] (default)
/NOJOURNAL

Determines whether VAXTPU keeps a journal file of your editing session, so you can recover your work in case of a system failure. VAXTPU applications create the journal file in the current (default) directory. By default, EVE maintains a journal file that has the same name as the input file and the file type TJL. If you invoke VAXTPU without an input file specification, the default name for the journal file is TPU.TJL. To prevent VAXTPU from keeping a journal file for your editing session, use the qualifier /NOJOURNAL.

/MODIFY
/NOMODIFY

Determines whether the first buffer in an editing session is modifiable. If you do not specify either form of the qualifier, VAXTPU determines the buffer status from the /READ_ONLY or /WRITE qualifier. By default, a read-only buffer is unmodifiable and a write buffer is modifiable.

/OUTPUT=output-file (default)
/NOOUTPUT

Determines the specification of the output file that is created at the end of your editing session. Processing this qualifier depends on the VAXTPU application you are using. By default, EVE uses the same file specification for both the input file and the output file. The output file has a version number one higher than the highest existing version of the input file. To specify an output file different from the input file (such as to write the output in a diferent directory or device), use /OUTPUT= and specify the output file you want. Using /NOOUTPUT suppresses the creation of an output file, but does not affect creation of a journal file. If you use /NOOUTPUT, EVE makes the main or first buffer in an edit session a no-write buffer. If you

enter /NOOUTPUT and then decide you want an output file, use the EVE command WRITE FILE before ending the editing session to write out the contents of a buffer.

/READ_ONLY
/NOREAD_ONLY (default)
Determines whether VAXTPU creates an output file from the contents of the main buffer. With /NOREAD_ONLY, VAXTPU creates an output file from the contents of the main buffer if you modified it. Using the qualifier /READ_ONLY is effectively the same as using /NOOUTPUT for the main buffer. Typically, you use /READ_ONLY for demonstration sessions or to examine a file without making any edits. (If you do make edits and want to save them, use the EVE command WRITE FILE before ending the session.)

You cannot specify the combination of /READ_ONLY and /WRITE, or the combination of /NOREAD_ONLY and /NO_WRITE on the same command line. VAXTPU signals an error and returns control to DCL if it encounters either of these combinations.

/RECOVER
/NORECOVER (default)
Determines whether VAXTPU reads a journal file at the start of an editing session to recover your work from an interrupted session. To recover your edits, reenter the command for the original, interrupted session, including all qualifiers, and add /RECOVER. If you originally used /JOURNAL= and specified a journal file other than the default (for example, if you specified a different directory for the journal file or a file type other than TJL), you must also use /JOURNAL= and specify the journal file.

When you recover a session, all files must be in the same state as at the start of the editing session being recovered. All terminal characteristics must also be in the same state as at the start of that session. Carefully check the following terminal characteristics:

Device_Type
Edit_mode
Eightbit
Page
Width

/SECTION[=file-spec] (default)
/NOSECTION
Determines whether VAXTPU reads a section file, containing in binary form, key definitions and compiled procedures. The default file type for section files is TPU$SECTION. By default, VAXTPU tries to read the file SYS$SHARE:TPU$SECTION.TPU$SECTION, which is the EVE section file.

To specify a different section file, define TPU$SECTION or use /SECTION= and specify the section file you want. Use a full file specification for the section file, including the device (or disk) and directory. Otherwise, VAXTPU assumes it is in SYS$SHARE. To create a section file, use the built-in procedure SAVE or the EVE command SAVE EXTENDED TPU. If you do not define TPU$SECTION or do not specify a section file with /SECTION= on the command line, VAXTPU defines TPU$SECTION to point to SYS$SHARE:EVE$SECTION.TPU$SECTION.

If you specify /NOSECTION, VAXTPU does not read a section file. If no section file is read, and no command file is read, VAXTPU will not have a user interface and no keys will be defined. In this state, the only way to exit from VAXTPU is to press CTRL/Y.

/START_POSITION=(row, column) (default= 1,1)
Determines the row and column in the main buffer where the cursor first appears. By default, the start position is the first row, first column (upper left corner) of the buffer.

/WRITE (default)
/NOWRITE
Determines whether you can edit text in the first buffer in a session. If you specify /NOWRITE, VAXTPU creates a read-only buffer for the first input file. VAXTPU does not create a new output file for such a buffer. This qualifier does not affect buffers other than the first buffer created during the session.

If you specify /WRITE or if you do not specify any qualifier related to /READ_ONLY or /WRITE, the first buffer is a write buffer. The /WRITE qualifier is the opposite of the /READ_ONLY qualifier.

You cannot specify the combination of /WRITE and /READ_ONLY, or the combination and /NO_WRITE and /NOREAD_ONLY on the same command line. VAXTPU signals an error and returns control to DCL if it encounters either of these combinations.

example

```
$ EDIT/TPU/OUTPUT=newfile.txt oldfile.txt
```

The EDIT/TPU command in this example invokes VAXTPU (running EVE) to edit the file OLDFILE.TXT. Modifying the main buffer and then using the command EXIT to end the session creates an output file called NEWFILE.TXT.

EOD

Signals the end of a data stream when a command or program is reading data from an input device other than an interactive terminal. Used to end a data line that begins with a dollar sign. Also, used to end an input file if more than one input file is contained in the command stream without intervening commands.

format

$ EOD

parameters

None.

example

```
$ CREATE WEATHER.COM
$ DECK
$ FORTRAN WEATHER
$ LINK WEATHER
$ RUN WEATHER
$ EOD
$ @WEATHER
```

In this example, the command procedure creates a command procedure called WEATHER.COM. The lines delimited by the DECK and EOD commands are written to the file WEATHER.COM. Then the command procedure executes WEATHER.COM.

EOJ

Marks the end of a batch job submitted through a card reader. An EOJ card is not required; however, if present, the first nonblank character in the command line must be a dollar sign ($). If issued in any other context, the EOJ command logs the process out. The EOJ command cannot be abbreviated.

The EOF card is equivalent to the EOJ card.

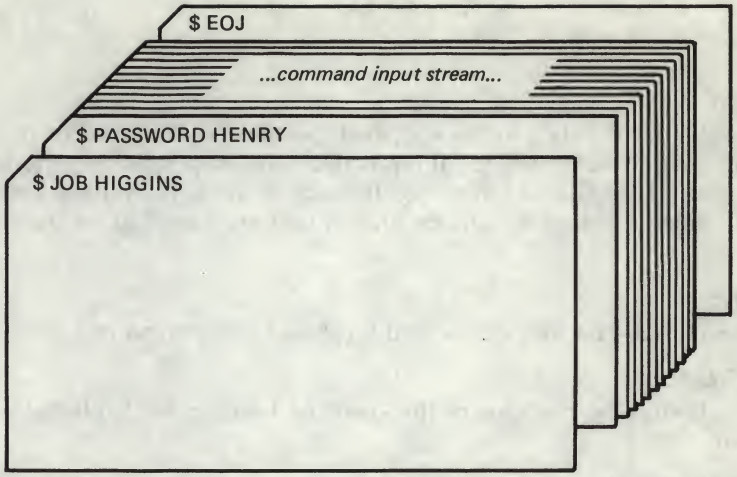
format

$ EOJ

parameters

None.

example



ZK-786-82

The JOB and PASSWORD commands mark the beginning of a batch job submitted through the card reader; the EOJ command marks the end of the job.

EXAMINE

Displays the contents of virtual memory.

Requires user mode read (R) and write (W) access to the virtual memory location whose contents you want to examine.

format

EXAMINE *location[:location]*

parameter

location[:location]
Specifies a virtual address or a range of virtual addresses (where the second address is larger than the first) whose contents you want to examine. If you specify a range of addresses, separate the first and last with a colon. A location can be any valid arithmetic expression containing arithmetic or logical operators or previously assigned symbols. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial

default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X).

qualifiers

/ASCII

Requests that the data at the specified location be displayed in ASCII. Binary values that do not have ASCII equivalents are displayed as periods (.). When you specify /ASCII, or when ASCII mode is the default, hexadecimal is used as the default radix for numeric literals that are specified on the command line.

/BYTE

Requests that data at the specified location be displayed one byte at a time.

/DECIMAL

Requests that the contents of the specified location be displayed in decimal format.

/HEXADECIMAL

Requests that the contents of the specified location be displayed in hexadecimal format.

/LONGWORD

Requests that data at the specified location be displayed one longword at a time.

/OCTAL

Requests that the contents of the specified location be displayed in octal format.

/WORD

Requests that data at the specified location be displayed one word at a time.

example

```
$ RUN   MYPROG
CTRL/Y
$ EXAMINE   2678
0002678:  1F4C5026
$ CONTINUE
```

In this example, the RUN command begins execution of the image MYPROG.EXE. While MYPROG is running, CTRL/Y interrupts its execution, and the EXAMINE command requests a display of the contents of virtual memory location 2678 (hexadecimal).

EXCHANGE

Invokes the Exchange Utility (EXCHANGE) to manipulate mass storage volumes that are written in formats other than those normally recognized by the VMS operating system.

EXCHANGE allows you to perform any of the following tasks:

- Create foreign volumes
- Transfer files to and from the volume
- List directories of the volume

For block-addressable devices, such as RT–11 disks, EXCHANGE performs additional operations such as renaming and deleting files. The Exchange Utility can also manipulate Files–11 files that are images of foreign volumes; these files are called *virtual devices*.

The /[NO]MESSAGE qualifier determines whether EXCHANGE displays information related to EXCHANGE INITIALIZE, MOUNT, and DISMOUNT subcommands. You can also use this qualifier with any of these three subcommands to reverse the default. Normally, EXCHANGE displays the information.

For more information about the Exchange Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

EXIT

Terminates processing of a command procedure and returns control to the next higher command level — either an invoking command procedure or DCL. The EXIT command also terminates an image normally after a user enters CTRL/Y (executing another image has the same effect).

format

EXIT *[status-code]*

parameter

status-code

Defines a numeric value for the reserved global symbol $STATUS. You can specify the status-code as an integer or an expression equivalent to an integer value. The low-order three bits of the value determine the value of the global symbol $SEVERITY.

example

```
$ ON WARNING THEN EXIT
$ FORTRAN 'P1'
$ LINK 'P1'
$ RUN 'P1'
```

The EXIT command in this example is used as the target of an ON command; this statement ensures that the command procedure terminates whenever any warnings or errors are issued by any command in the procedure.

The procedure exits with the status value of the command or program that caused the termination.

GOSUB

Transfers control to a labeled subroutine in a command procedure without creating a new procedure level.

format

GOSUB *label*

parameter

label

Specifies a 1- through 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the GOSUB command is executed, control passes to the command following the specified label. The label can precede or follow the GOSUB statement in the current command procedure. When you use a label in a command procedure, it must be terminated with a colon. If you use duplicate labels, control is always given to the label most recently read by DCL.

example

```
$!
$! GOSUB.COM
$!
$ SHOW TIME
$ GOSUB TEST1
$ WRITE SYS$OUTPUT "success completion"
$ EXIT
$!
$! TEST1 GOSUB definition
$!
$ TEST1:
$     WRITE SYS$OUTPUT "This is GOSUB level 1."
$     GOSUB TEST2
$     RETURN %X1
$!
$! TEST2 GOSUB definition
$!
$ TEST2:
$     WRITE SYS$OUTPUT "This is GOSUB level 2."
$     GOSUB TEST3
$     RETURN
$!
$! TEST3 GOSUB definition
$!
$ TEST3:
$     WRITE SYS$OUTPUT "This is GOSUB level 3."
$     RETURN
```

This sample command procedure shows how to use the GOSUB command to transfer control to labeled subroutines. The GOSUB command transfers control to the subroutine labeled TEST1. The procedure executes the commands in subroutine TEST1, branching to the subroutine labeled TEST2. The procedure then executes the commands in subroutine TEST2, branching to the subroutine labeled TEST3. Each subroutine is terminated by the RETURN command. After TEST3 is executed, the RETURN command returns control back to the command line following each calling GOSUB statement. At this point, the procedure has been successfully executed.

GOTO

Transfers control to a labeled statement in a command procedure.

format

GOTO *label*

parameter

label

Specifies a 1- through 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the GOTO command is executed, control passes to the command following the specified label. The label can precede or follow the GOTO statement in the current command procedure. When you use a label in a command procedure, it must be terminated with a colon. If you use duplicate labels, control is always given to the label most recently read by DCL.

example

```
$ IF P1 .EQS. "HELP" THEN GOTO TELL
$ IF P1 .EQS. "" THEN GOTO TELL
   .
   .
   .
$ EXIT
$ TELL:
$ TYPE SYS$INPUT
To use this procedure, you must enter a value for P1.
   .
   .
   .
$ EXIT
```

In this example, the IF command checks the first parameter passed to the command procedure; if this parameter is the string HELP or if the parameter is not specified, the GOTO command is executed and control is passed to the line labeled TELL. Otherwise, the procedure continues executing until the EXIT command is encountered. At the label TELL, a TYPE command displays data in the input stream that documents how to use the procedure.

HELP

Displays information concerning use of the system, including formats and explanations of commands, parameters, and qualifiers.

format

HELP *[keyword ...]*

parameter

keyword ...

Specifies one or more keywords that refer to the topic or subtopic on which you want information from a HELP library. To use the HELP facility in its simplest form, enter the HELP command from your terminal. The HELP facility displays a list of topics at your terminal and the prompt Topic?. To see information on one of the topics, type the topic name after the prompt. The system displays information on that topic.

If the topic has subtopics, the HELP command lists the subtopics and displays the Subtopic? prompt. To get information on one of the subtopics, type the name after the prompt. To see information on another topic, press RETURN. You can now ask for information on another topic when HELP displays the Topic? prompt. Press RETURN to exit the HELP facility and return to the DCL command level.

example

```
$ HELP
HELP
  .
  .  (HELP message text and list of topics)
  .
Topic?
```

In this example, the HELP command is entered without any qualifiers or parameters. This produces a display of the HELP topics available from the root HELP library, SYS$HELP:HELPLIB.HLB.

If you type one of the listed topics in response to the Topic? prompt, HELP displays information about that topic and a list of subtopics (if there are any). If one or more subtopics exist, HELP prompts you for a subtopic.

```
Topic? ASSIGN
ASSIGN
  .
  .  (HELP message text and subtopics)
  .
ASSIGN Subtopic?
```

If you type a subtopic name, HELP displays information about that subtopic:

```
ASSIGN Subtopic? Name
ASSIGN
  Name
  .
  .  (HELP message text and subtopics, if any)
  .
ASSIGN Subtopic?
```

If one or more sub-subtopics exist, HELP prompts for a sub-subtopic; otherwise, as in the previous example, the facility prompts you for another subtopic of the topic you are currently inspecting.

Typing a question mark redisplays the HELP message and options at your current level. Pressing RETURN does either of the following: (1) move you back to the previous HELP level if you are in a subtopic level, or (2) terminate HELP if you are at the first level. Pressing CTRL/Z terminates HELP at any level.

IF

Tests the value of an expression and, depending on the syntax specified, executes

- One command following the THEN keyword if the expression is true
- Multiple commands following the $THEN command if the expression is true
- One or more commands following the $ELSE command if the expression is false

format

$ IF *expression THEN [$] command*

or

$ IF *expression*
$ THEN *[command]*
command
.
.

$ [ELSE] *[command]*
command
.
.
$ ENDIF

parameters

expression
Defines the test to be performed. The expression can consist of one or more numeric constants, string literals, symbolic names, or lexical functions separated by logical, arithmetic, or string operators. Expressions in IF commands are automatically evaluated during the execution of the command. Character strings beginning with alphabetic characters that are not enclosed in quotation marks are assumed to be symbol names or lexical functions. The Command Language Interpreter (CLI) replaces these strings with their current values. Symbol substitution in expressions in IF commands is not iterative; that is, each symbol is replaced only once. However, if you want iterative substitution, precede a symbol name with an apostrophe or ampersand.

command
The DCL command or commands to be executed, depending on the syntax specified, when the result of the expression is true or false.

example

```
$ IF P1 .EQS. "" THEN GOTO DEFAULT
$ IF (P1 .EQS. "A") .OR. (P1 .EQS. "B") THEN GOTO 'P1'
$  WRITE SYS$OUTPUT "Unrecognized parameter option ''P1' "
$ EXIT
$  A:       !  Process option a
 .
 .
 .
$ EXIT
$  B:       !  Process option b
 .
 .
 .
$ EXIT
$  DEFAULT: !  Default processing
 .
 .
 .
$ EXIT
```

This example shows a command procedure that tests whether a parameter was passed. The GOTO command passes control to the label specified as the parameter.

If the procedure is executed with a parameter, the procedure uses that parameter to determine the label to branch to. For example:

```
@TESTCOM A
```

When the procedure executes, it determines that P1 is not null, and branches to the label A. Note that the EXIT command causes an exit from the procedure before the label B.

```
$  SET NOON
 .
 .
 .
$  LINK CYGNUS,DRACO,SERVICE/LIBRARY
$ IF $STATUS
$ THEN
$  RUN CYGNUS
$ ELSE
$   WRITE SYS$OUTPUT "LINK FAILED"
$ ENDIF
$ EXIT
```

This command procedure uses the SET NOON command to disable error checking by the command procedure. After the LINK command, the IF command tests the value of the reserved global symbol $STATUS. If the value of $STATUS indicates that the LINK command succeeded, then the program CYGNUS is run. If the LINK command returns an error status value, the command procedure issues a message and exits.

INITIALIZE

Formats a disk or magnetic tape volume and writes a label on the volume. At the end of initialization, the disk is empty except for the system files containing the structure information. All former contents of the disk are lost.

Requires VOLPRO privilege for most INITIALIZE operations.

format

INITIALIZE *device-name[:] volume-label*

parameters

device-name[:]
Specifies the name of the device on which the volume to be initialized is physically mounted.

volume-label
Specifies the identification to be encoded on the volume. For a disk volume, you can specify a maximum of 12 alphanumeric characters; for a magnetic tape volume, you can specify a maximum of 6 alphanumeric characters.

Letters are automatically changed to uppercase. Nonalphanumeric characters are not allowed in the volume-label specification on disk.

qualifiers

/ACCESSED=number-of-directories

Requires OPER privilege. Affects Files-11 Structure Level 1 disks ONLY. Specifies, for disk volumes, the number of directories allowed in system space must be a value from 0 to 255. The default value is 3.

/BADBLOCKS=(area[,...])

Specifies, for disk volumes, faulty areas on the volume. The INITIALIZE command marks the areas as allocated so that no data is written in them.

Possible formats for area are as follows:

lbn[:count]	Logical block number of the first block and optionally a block count beginning with the first block, to be marked as allocated
sec.trk.cyl[:cnt]	Sector, track, and cylinder of the first block, and optionally a block count beginning with the first block, to be marked as allocated

/CLUSTER_SIZE=number-of-blocks

Defines, for disk volumes, the minimum allocation unit, in blocks. The maximum size you can specify for a volume is one-hundredth the size of the volume; the minimum size you can specify is calculated with the following formula:

$$\frac{disk\ size(number\ of\ blocks)}{255 * 4096}$$

/DATA_CHECK[=(option[,...])]

Checks all read and write operations on the disk. By default, no data checks are made. Specify one or both options:

READ	Checks all read operations
WRITE	Checks all write operations; default if only /DATA_CHECK is specified

To override the checking you specify at initialization for disks, enter a MOUNT command to mount the volume.

/DENSITY=density-value

The /DENSITY qualifier is not applicable to the TK50 tape device. For floppy disk volumes that are to be initialized on RX02 dual-density disk drives, specifies the density at which the floppy disk is to be formatted.

For magnetic tape volumes, specifies the density in bytes per inch (bpi) at which the magnetic tape is to be written.

RX02 dual-density disk drives allow floppy disks to be initialized at single or double density. To specify single-density formatting of a floppy disk, specify the density value SINGLE. To specify double-density formatting of a floppy disk, specify the density value DOUBLE.

If you do not specify a density value for a floppy disk being initialized on an RX02 drive, the system leaves the volume at the density to which the volume was last formatted. Floppy disks purchased from DIGITAL are formatted in single density.

For magnetic tape volumes, the density value specified can be 800 bpi, 1600 bpi, or 6250 bpi, as long as the density is supported by the magnetic tape drive.

/DIRECTORIES=number-of-entries
Specifies, for disk volumes, the number of entries to preallocate for user directories. The number of entries must be an integer between 16 and 16000. The default value is 16.

/ERASE
/NOERASE (default)
Physically destroys deleted data (by writing over it).

/EXTENSION=number-of-blocks
Affects Files-11 Structure Level 1 disks ONLY. Specifies, for disk volumes, the number of blocks to use as a default extension size for all files on the volume. The extension default is used when a file increases to a size greater than its initial default allocation during an update. The value for the number-of-blocks parameter can range from 0 through 65,535. The default value is 5.

/FILE_PROTECTION=code
Affects Files-11 Structure Level 1 disks ONLY. Defines, for disk volumes, the default protection to be applied to all files on the volume.

/GROUP
Defines a group volume. The /GROUP qualifier applies protection of RWED to all ownership categories unless /GROUP is specified with /NOSHARE, in which case the volume protection is RWED for all but the world category. The owner UIC of the volume defaults to your group number and a member number of 0.

/HEADERS=number-of-headers
Specifies, for disk volumes, the number of file headers to be allocated for the index file. The minimum and default value is 16. The maximum is the value set with the /MAXIMUM_FILES qualifier.

/HIGHWATER (default)
/NOHIGHWATER
Affects Files-11 Structure Level 2 disks ONLY. Sets the file highwater mark (FHM) volume attribute, which guarantees that a user cannot read data that he has not written. You cannot specify /NOHIGHWATER for magnetic tape.

The /NOHIGHWATER qualifier disables FHM for a disk volume.

/INDEX=position

Specifies the location of the index file for the volume's directory structure. Possible positions are as follows:

BEGINNING	Beginning of the volume
MIDDLE	Middle of the volume (default)
END	End of the volume
BLOCK:n	Beginning of the logical block specified by *n*

/LABEL=option

Defines characteristics for the magnetic tape volume label, as directed by the included option. The available options are as follows:

- OWNER_IDENTIFIER:"(14 ANSI characters)"

  Allows you to specify the Owner Identifier field in the volume label. The field specified can accept up to 14 ANSI characters.

- VOLUME_ACCESSIBILITY:"character"

  Specifies the character to be written in the volume accessibility field of the VMS ANSI volume label VOL1 on an ANSI magnetic tape. The character may be any valid ANSI "a" character. This set of characters includes numeric characters, uppercase letters, and any one of the following nonalphanumeric characters:

  ! " % ' ( ) * + , - . / : ; < = > ?

  By default, the VMS operating system provides a routine that checks this field in the following manner.

  - If the magnetic tape was created on a version of the VMS operating system that conforms to Version 3 of ANSI, then this option must be used to override any character other than an ASCII space.
  - If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, then this option must be used to override any character other than an ASCII 1.

  If you specify any character other than the default, you must specify the /OVERRIDE=ACCESSIBILITY qualifier on the INITIALIZE and MOUNT commands in order to access the magnetic tape.

/MAXIMUM_FILES=n

Restricts the maximum number of files that the volume can contain. The /MAXIMUM_FILES qualifier overrides the default value, which is calculated as follows:

$$\frac{volume\ size\ in\ blocks}{(cluster\ factor + 1) * 2}$$

The maximum size you can specify for any volume is as follows:

$$\frac{volume\ size\ in\ blocks}{(cluster\ factor + 1)}$$

The minimum value is 0. Note that the maximum can be increased only by reinitializing the volume.

/OVERRIDE=(option[,...])

Requests the INITIALIZE command to ignore data on a magnetic tape volume that protects it from being overwritten. You may specify one or more of the following options:

ACCESSIBILITY	(For magnetic tapes only.) If the installation allows, this option overrides any character in the Accessibility Field of the volume. The necessity of this option is defined by the installation. That is, each installation has the option of specifying a routine that the magnetic tape file system will use to process this field. By default, VMS provides a routine that checks this field in the following manner. If the magnetic tape was created on a version of VMS that conforms to Version 3 of ANSI, this option must be used to override any character other than an ASCII space. If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, this option must be used to override any character other than an ASCII 1. To use the ACCESSIBILITY option, you must have the user privilege VOLPRO or be the owner of the volume.
EXPIRATION	(For magnetic tapes only.) Allows you to write to a tape that has not yet reached its expiration date. You must have the user privilege VOLPRO to override volume protection, or your UIC must match the UIC written on the volume.
OWNER_IDENTIFIER	Allows you to override the processing of the Owner Identifier field of the volume label.

To initialize a volume that was initialized previously with the /PROTECTION qualifier, your UIC must match the UIC written on the volume or you must have VOLPRO privilege.

/OWNER_UIC=uic

Specifies an owner UIC for the volume. The default is your default UIC.

For magnetic tapes, no UIC is written unless protection on the magnetic tape is specified. If protection is specified, but no owner UIC is specified, your current UIC is assigned ownership of the volume.

/PROTECTION=(ownership[:access],...)

Applies the specified protection to the volume. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (read), W (write), E (execute), and D (delete). The default is your default protection.

For magnetic tape, the protection code is written to a VMS-specific volume label. The system only applies read and write access restrictions; execute and delete access are meaningless. Moreover, the system and the owner are always given both read and write access to magnetic tapes, regardless of the protection code you specify.

/SHARE (default)
/NOSHARE

Permits all categories of access by all categories of ownership. The /NOSHARE qualifier denies access to group (unless /GROUP is also specified) and world processes.

/STRUCTURE=level

Specifies whether the volume should be formatted in Files–11 Structure Level 1 or Structure Level 2 (the default). Level 1 is incompatible with the /DATA_CHECK and /CLUSTER_SIZE qualifiers. The default protection for a Structure Level 1 disk is full access to system, owner, and group, and R (read) access to all other users.

/SYSTEM

Requires a system UIC or SYSPRV privilege. Defines a system volume. The owner UIC defaults to [1,1]. Protection defaults to complete access by all ownership categories, except that only system processes can create top-level directories.

/USER_NAME=name

Specifies a user name to be associated with the volume. The name must be 1 to 12 alphanumeric characters. The default is your user name.

/VERIFIED
/NOVERIFIED

Indicates whether the disk has bad block data on it. Use the /NOVERIFIED qualifier to ignore bad block data on the disk. The default is /VERIFIED for disks with 4096 blocks or more and /NOVERIFIED for disks with less than 4096 blocks.

/WINDOWS=n

Specifies the number of mapping pointers (used to access data in the file) to be allocated for file windows. The value can be an integer in the range of 7 through 80. The default is 7.

example

```
$ INITIALIZE/USER_NAME=CPA $FLOPPY1 ACCOUNTS
```

Initializes the volume on $FLOPPY1, labels the volume ACCOUNTS, and gives the volume a user name of CPA.

INITIALIZE/QUEUE

Creates or initializes queues. You use this command to create queues and to assign them names and attributes.

Requires OPER privilege. Requires the /BATCH qualifier to create a batch queue.

format

INITIALIZE/QUEUE *queue-name[:]*

parameter

queue-name[:]
Specifies the name of an execution queue or a generic queue. The queue name may be up to 31 alphanumeric characters.

qualifiers

/BASE_PRIORITY=n
Specifies the base process priority at which jobs are initiated from a batch queue or the base priority of the symbiont process for a printer, terminal, or server queue. By default, if you omit the qualifier, jobs are initiated at the same priority as the base priority established by DEFPRI at system generation. The n specifier can be any decimal value from 0 to 15.

/BATCH
/NOBATCH (default)
Specifies that you are initializing a batch queue. If you are reinitializing an existing queue, you can use the /BATCH qualifier only if the queue was created as a batch queue. A batch queue is classified as either an execution or generic queue. By default, the /BATCH qualifier initializes an execution queue. To specify a generic batch queue, use the /GENERIC qualifier together with the /BATCH qualifier. The /BATCH and /DEVICE qualifiers are mutually exclusive; the /NOBATCH and /NODEVICE qualifiers also cannot be used together.

/BLOCK_LIMIT=([lowlim,]uplim)
/NOBLOCK_LIMIT (default)
Limits the size of print jobs that can be executed on a printer or terminal queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters. The lowlim parameter is a decimal number referring to the minimum number of blocks accepted by the queue for a print job. If a print job is submitted that contains fewer blocks than the lowlim value, the job remains pending until the block limit for the queue is changed, enabling it to execute. The uplim parameter is a decimal number referring to the maximum number of blocks that will be accepted by the queue for a print job. If a print job is submitted that exceeds

this value, the job remains pending until the block limit for the queue is changed, enabling it to execute.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS (default)

Specifies one or more characteristics for processing jobs on the queue. If only one characteristic is specified, you can omit the parentheses. Each time you specify /CHARACTERISTICS, all previously set characteristics are erased. Only the ones specified with the qualifier are now established for the queue. Queue characteristics are installation-specific. The characteristic parameter can be either a value from 0 to 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.

/CLOSE

Prevents jobs from being entered in the queue through PRINT or SUBMIT commands or as a result of requeue operations. To allow jobs to be entered, use the /OPEN qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled). When a queue is marked closed, jobs executing continue to execute, and jobs already pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Indicates the default CPU time limit for batch jobs. Time can be specified as a delta time, 0, NONE (the default), or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time (subject to the restrictions imposed by the /CPUMAXIMUM qualifier or the user authorization file).

/CPUMAXIMUM=time

Indicates the maximum CPU time limit for batch jobs. The /CPUMAXIMUM qualifier overrides the time limit specified in the user authorization file (UAF). Time can be specified as a delta time, 0, NONE (the default), or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time. Specify NONE when a maximum CPU time limit is not desired.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. Once an option is set for the queue by the /DEFAULT qualifier, users do not have to specify that option in their PRINT commands. The /DEFAULT qualifier cannot be used with the /GENERIC qualifier. Possible options are as follows:

[NO]BURST[=keyword]	Specifies where to print burst pages (flag pages that are printed over the paper's perforations for easy identification of individual files in a print job). The keyword ALL places burst pages before each printed file in the job. The keyword ONE places a burst page before the first printed file in the job.
[NO]FEED	Specifies whether a form feed is automatically inserted at the end of a page. (The default is FEED.)
[NO]FLAG[=keyword]	Specifies where to print flag pages (containing the job entry number, the name of the user submitting the job, and so on). The keyword ALL places flag pages before each printed file in the job. The keyword ONE places a flag page before the first printed file in the job.
FORM=type	Specifies the default form for a printer, terminal, or server queue. If a job is not submitted with an explicit form definition, then this form will be used to process the job. The systemwide default form, form=0, is the default value for this keyword. See also /FORM_MOUNTED.
[NO]TRAILER[=keyword]	Specifies where to print trailer pages. The keyword ALL places trailer pages after each printed file in the job. The keyword ONE places a trailer page after the last printed file in the job.

/DESCRIPTION=string
/NODESCRIPTION (default)

A string of up to 255 characters used to provide operator-supplied information about the queue.

If the string contains alphanumeric, underscore, or dollar sign characters it must be enclosed in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may have been associated with the queue.

/DEVICE[=option]
/NODEVICE

Specifies that you are initializing an output queue of a particular type. If you are reinitializing an existing queue, you can use the /DEVICE qualifier only if the queue was created as an output queue. Possible options are as follows:

PRINTER	Indicates that this is a printer queue.
SERVER	Indicates that this is a server queue. An execution server queue is controlled by the user-modified or user-written symbiont specified with the /PROCESSOR qualifier.
TERMINAL	Indicates that this is a terminal queue.

The use of /DEVICE without designating a queue type is equivalent to specifying /DEVICE=PRINTER. An output queue is classified as either an execution or generic queue. By default, the /DEVICE qualifier initializes an execution queue of the designated type. To specify a generic printer, server, or terminal queue, use the /GENERIC qualifier together with the /DEVICE qualifier. For an output execution queue, the queue type you specify with

the /DEVICE qualifier is for informational purposes. When the queue is started, the symbiont associated with the queue determines the actual queue type. The /DEVICE and /BATCH qualifiers are mutually exclusive; the /NODEVICE and /NOBATCH qualifiers also cannot be used together.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING (default)

Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC (default)
/NOENABLE_GENERIC

Specifies whether files queued to a generic queue that does not have specific associated execution queues (named with the /GENERIC qualifier) can be placed in this execution queue for processing. (See the description of the /GENERIC qualifier for more information.)

/FORM_MOUNTED=type

Specifies the form type for a printer, terminal, or server queue. If the stock of the mounted form is not identical to the stock of the default form, as indicated by the DCL command qualifier /DEFAULT=FORM=type, all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, the job enters a pending state. In both cases, the pending state is maintained until the stock of the mounted form of the queue is identical to the stock of the form associated with the job. Specify the form type using either a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific. The /FORM_MOUNTED qualifier is incompatible with the /GENERIC qualifier.

/GENERIC[=(queue-name[,...])]
/NOGENERIC (default)

Specifies that this is a generic queue and that jobs placed in it can be moved for processing to compatible execution queues. The /GENERIC qualifier optionally accepts a list of target execution queues that have been previously defined. For a generic batch queue, these target queues must be batch execution queues. For a generic output queue, these target queues must be output execution queues, but can be of any type (printer, server, or terminal). If you do not specify any target queues with the /GENERIC qualifier, jobs can be moved to any execution queue that (1) is initialized with the /ENABLE_GENERIC qualifier, and (2) is the same type (batch, printer, server, or terminal) as the generic queue. Moreover, for a generic server queue, an additional check is made: the symbiont named with the /PROCESSOR qualifier must be the same for both the generic and execution queues. The /GENERIC qualifier is used in conjunction with either the /BATCH or /DEVICE qualifiers to define the queue as a generic batch, printer, server, or terminal queue. If neither /BATCH nor /DEVICE is

specified on creation of a generic queue, it becomes a generic printer queue by default.

/JOB_LIMIT=n

Indicates the number of batch jobs that can be executed concurrently from the queue. Specify a number in the range 0 to 255. The job limit default value for n is 1.

/LIBRARY=file-name
/NOLIBRARY

Specifies the file name for the device control library. When you are initializing an output queue, you can use the /LIBRARY qualifier to specify an alternate device control library. The default library is SYS$LIBRARY:SYSDEVCTL.TLB. Only a file name can be used as the parameter of the /LIBRARY qualifier. The system always assumes that the location of the file is in SYS$LIBRARY and that the file type is TLB.

/ON=[node::]device[:] (printer, terminal, server queue)
/ON=node:: (batch queue)

Specifies the node or device, or both, on which this execution queue is located. For batch queues, only the node name can be specified. You can include both the node name and the device name for printer and terminal queues. By default, a queue executes on the same node from which you first start the queue. The default device parameter is the same as the queue name.

/OPEN (default)

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled).

/OWNER_UIC=uic

Enables you to change the UIC of the queue. The default UIC is [1,4].

/PROCESSOR=file-name
/NOPROCESSOR

Allows users to specify their own print symbionts. The file name specifier can be any valid file name. Only a file name can be used as a parameter of the /PROCESSOR qualifier. The system supplies the device and directory name SYS$SYSTEM as well as the file type EXE.

/PROTECTION=(codes)

Specifies the protection of the queue. Ownership categories are SYSTEM, OWNER, GROUP, WORLD; each category can be abbreviated to its first character. Access categories are R (READ), W (WRITE), E (EXECUTE), or D (DELETE); a null access specification means no access. The default protection is: (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W).

/RECORD_BLOCKING (default)
/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify /NORECORD_BLOCKING, the symbiont is directed to send each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN (default)

Holds jobs in the queue in a completed status after they have executed. The /NORETAIN qualifier enables you to reset the queue to the default. Possible options are as follows:

ALL (default)	Holds all jobs in the queue after execution
ERROR	Holds in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in a printer, terminal, or server queue are scheduled for printing based on the size of the job. When the default, /SCHEDULE=SIZE, is in effect, shorter jobs print before longer ones.

Note that if you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEPARATE=(option[,...])
/NOSEPARATE (default)

Specifies the job separation defaults for a printer or terminal queue. The /SEPARATE qualifier is incompatible with the /GENERIC qualifier. The job separation options are as follows:

[NO]BURST	Specifies whether a burst page prints at the beginning of every job. Specifying BURST also results in a flag page being printed.
[NO]FLAG	Specifies whether a flag page prints at the beginning of every job.
[NO]TRAILER	Specifies whether a trailer page prints at the end of every job.
[NO]RESET=(module[,...])	Specifies a job reset sequence for the queue. The specified modules from the device control library are used to reset the device each time a job reset occurs.

/START
/NOSTART (default)

Starts the queue being initialized by the current INITIALIZE/QUEUE command.

/TERMINAL
/NOTERMINAL (default)

Indicates that the output queue is a terminal queue. The /NOTERMINAL qualifier cancels the effect of a previous /TERMINAL qualifier on the same command. It is supported in this release for compatibility with VMS V4.n. The function of the /[NO]TERMINAL qualifier has been superseded by the /[NO]DEVICE qualifier. DIGITAL recommends that you use this new qualifier and that existing command procedures using /[NO]TERMINAL be updated.

/WSDEFAULT=n

Defines a working set default for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values are a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set default value defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

/WSEXTENT=n

Defines a working set extent for the batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values are a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set value defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

/WSQUOTA=n

Defines the working set page size (working set quota) for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values are a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set quota defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

example

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG SYS$PRINT/ON=LPA0:
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=4 SYS$BATCH
```

In this example, the two commands initialize and start the printer queue SYS$PRINT on device LPA0 and then the batch queue SYS$BATCH. The /DEFAULT=FLAG qualifier causes a flag page to precede each file for jobs in the printer queue. The /JOB_LIMIT=4 qualifier allows as many as four batch jobs to be initiated concurrently from the batch queue. Both queues are started as soon as they have been initialized.

INQUIRE

Reads a value from SYS$COMMAND (usually the terminal in interactive mode or the next line in the main command procedure) and assigns it to a symbol.

format

INQUIRE *symbol-name [prompt-string]*

parameters

symbol-name

Specifies a 1- through 255-alphanumeric character symbol to be given a value.

prompt-string

Specifies the prompt to be displayed at the terminal when the INQUIRE command is executed. String values are automatically converted to uppercase. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space. Enclose the prompt in quotation marks (") if it contains lowercase characters, punctuation, multiple blanks or tabs, or an at sign (@). To denote an actual quotation mark in a prompt-string, enclose the entire string in quotation marks and use two consecutive quotation marks ("") within the string. If you do not specify a prompt string, the command interpreter uses the symbol name to prompt for a value.

qualifiers

/GLOBAL

Specifies that the symbol be placed in the global symbol table. If you do not specify the /GLOBAL qualifier, the symbol is placed in the local symbol table.

/LOCAL (default)

Specifies that the symbol be placed in the local symbol table for the current command procedure.

/PUNCTUATION (default)
/NOPUNCTUATION

Inserts a colon (:) and a space after the prompt when it is displayed on the terminal. To suppress the colon and space, specify /NOPUNCTUATION.

example

```
$ INQUIRE CHECK "Enter Y[ES] to continue"
$ IF .NOT. CHECK THEN EXIT
```

The INQUIRE command displays the following prompting message at the terminal:

```
Enter Y[ES] to continue:
```

The INQUIRE command prompts for a value, which is assigned to the symbol CHECK. The IF command tests the value assigned to the symbol CHECK. If the value assigned to CHECK is true (that is, an odd numeric value, a character string that begins with a T, t, Y, or y, or an odd numeric character string), the procedure continues executing.

If the value assigned to CHECK is false (that is, an even numeric value, a character string that begins with any letter except T, t, Y, or y, or an even numeric character string), the procedure exits.

INSTALL

Invokes the Install Utility, which enhances the performance of selected executable and shareable images by making them "known" to the system and assigning them appropriate attributes. For more information about the Install Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

JOB

Identifies the beginning of a batch job submitted through a card reader. Each batch job submitted through the system card reader must be preceded by a JOB card.

JOB cannot be abbreviated.

format

$ JOB *user-name*

parameter

user-name
Identifies the user name under which the job is to be run. Specify the user name as you would during the login procedure.

qualifiers

/AFTER=time

Holds the job until the specified time. If the specified time has already passed, the job is queued for immediate processing. The time can be specified as either an absolute time or a combination of absolute and delta times.

/CHARACTERISTICS=(characteristic[,...])

Specifies one or more characteristics required for processing the job. If you specify only one characteristic, you can omit the parentheses. Codes for characteristics are installation-defined. Use the SHOW QUEUE /CHARACTERISTICS command to see which characteristics are available on your system. All the characteristics specified for the job must also be specified for the queue that will execute the job. If not, the job remains pending in the queue until the queue characteristics are changed or the entry is deleted with the DELETE/ENTRY command. Users need not specify every characteristic of a queue with the JOB command as long as the ones they specify are a subset of the characteristics set for that queue. The job also runs if no characteristics are specified.

/CLI=file-name

Specifies a different command language interpreter (CLI) with which to process the job. The file name specifies that the CLI be SYS$SYSTEM:filename.EXE. The default CLI is that defined in the user authorization file (UAF).

/CPUTIME=n

Specifies a CPU time limit for the batch job. Time can be specified as delta time, 0, NONE, or INFINITE. Specify 0 or INFINITE to request an infinite amount of time. Specify NONE when you want the CPU time to default to your UAF value or the limit specified on the queue. Note that you cannot request more time than permitted by the base queue limits or your UAF.

/DELETE (default)
/NODELETE

Controls whether the batch input file is deleted after the job is processed. If you specify /NODELETE, the file is saved in the user's default directory under the default name INPBATCH.COM. If you specify the /NAME qualifier, the file name of the batch input file is the same as the job name you supply with /NAME.

/HOLD
/NOHOLD (default)

Controls whether or not the job is to be made available for immediate processing.

If you specify /HOLD, the job is not released for processing until you specifically release it with the /NOHOLD or /RELEASE qualifier of the SET QUEUE/ENTRY command.

/KEEP
/NOKEEP (default)
Controls whether the log file is deleted after it is printed. /NOKEEP is the default unless /NOPRINTER is specified.

/LOG_FILE=file-spec
/NOLOG_FILE
Controls whether a log file with the specified name is created for the job or whether a log file is created. When you use the /LOG_FILE qualifier, the system writes the log file to the file you specify. If you use /NOLOG_FILE, no log file is created. If you specify neither form of the qualifier, the log file is written to a file in your default directory that has the same file name as the first command file in the job and a file type of LOG. Using neither /LOG_FILE nor /NOLOG_FILE is the default. You can use the /LOG_FILE qualifier to specify that the log file be written to a different device.

/NAME=job-name
Specifies a file name string to be used as the job name and as the file name for both the batch job log file and the command file. The job name must be 1 to 39 alphanumeric characters and must be a valid file name. The default log file name is INPBATCH.LOG; the default command file name is INPBATCH.COM.

/NOTIFY
/NONOTIFY (default)
Controls whether a message is broadcast to any terminal at which you are logged in, notifying you when your job completes or aborts.

/PARAMETERS=(parameter[,...])
Specifies from 1 through 8 optional parameters that can be passed to the command procedure. The parameters define values to be equated to the symbols P1 through P8 in the batch job. The symbols are local to the specified command procedure. If you specify only one parameter, you can omit the parentheses. The commas delimit individual parameters. If the parameter contains any spaces, special characters or delimiters, or lowercase characters, enclose it in quotation marks. Individual parameters cannot exceed 255 characters.

/PRINTER=queue-name
/NOPRINTER
Controls whether the job log file is queued to the specified queue for printing when the job is complete. The default print queue for the log file is SYS$PRINT.

/PRIORITY=n

Requires OPER or alter priority (ALTPRI) privilege to raise the priority above the value of the SYSGEN parameter MAXQUEPRI. Specifies the job scheduling priority for the specified job. The value of *n* is an integer from 0 through 255, where 0 is the lowest priority and 255 is the highest. The default value for /PRIORITY is the value of the SYSGEN parameter DEFQUEPRI.

/QUEUE=queue-name[:]

Specifies the name of the batch queue in which the job is to be entered. If you do not specify /QUEUE, the job is placed in the default system batch job queue, SYS$BATCH.

/RESTART
/NORESTART (default)

Specifies whether the job restarts after a system failure or a STOP/QUEUE /REQUEUE command.

/TRAILING_BLANKS (default)
/NOTRAILING_BLANKS

Controls whether input cards in the card deck are read in card image form or input records are truncated at the last nonblank character. By default, the system does not remove trailing blanks from records read through the card reader.

/WSDEFAULT=n

Defines a working set default for the batch job; the /WSDEFAULT qualifier overrides the working set size specified in the user authorization file (UAF). *N* can be any positive integer from 1 to 65,535, 0, or the keyword NONE. A value of 0 or the keyword NONE sets the default value to the value specified either in your UAF or by the working set quota established for the queue. You cannot request a value higher than your default.

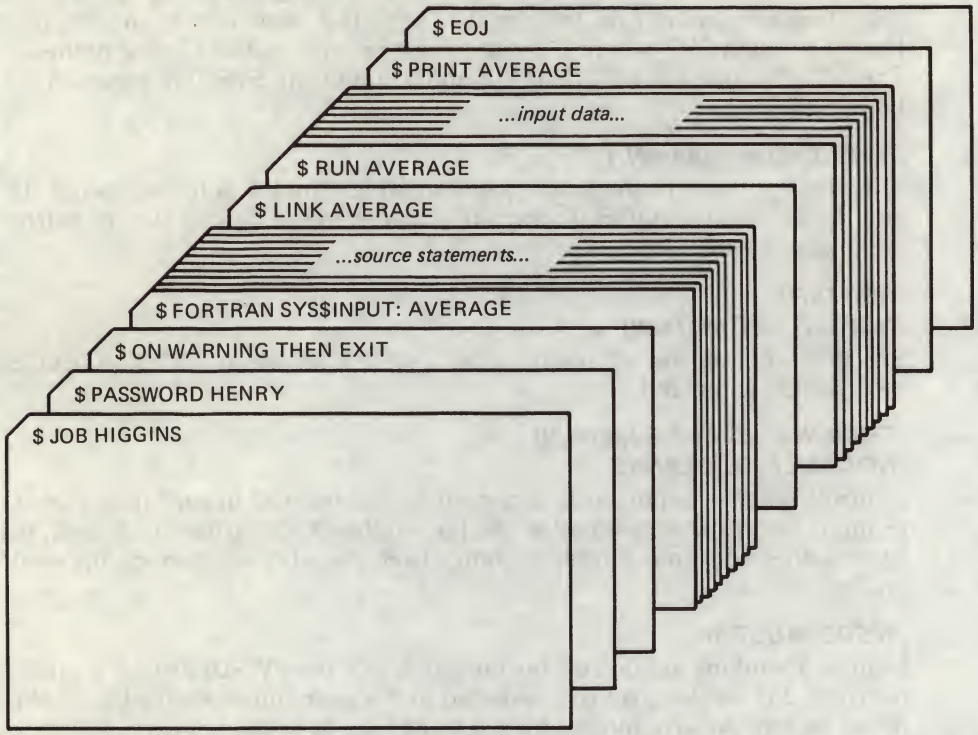
/WSEXTENT=n

Defines a working set extent for the batch job; the /WSEXTENT qualifier overrides the working set extent in the UAF. *N* can be any positive integer from 1 to 65,535, 0, or the keyword NONE. A value of 0 or the keyword NONE sets the default value either to the value specified in the UAF or working set extent established for the queue. You cannot request a value higher than your default.

/WSQUOTA=n

Defines the maximum working set size (working set quota) for the batch job; the /WSQUOTA qualifier overrides the value in the UAF. *N* can be any positive integer from 1 to 65,535, 0, or the keyword NONE. You cannot request a value higher than your default.

example



ZK-787-82

The JOB and PASSWORD cards identify and authorize the user HIGGINS to enter batch jobs. The command stream consists of a FORTRAN command and FORTRAN source statements to be compiled. The file name AVERAGE following the device name SYS$INPUT provides the compiler with a file name for the object and listing files. The output files are cataloged in user HIGGINS's default directory.

If the compilation is successful, the LINK command creates an executable image and the RUN command executes it. Input for the program follows the RUN command in the command stream. The last command in the job prints the program listing. The last card in the deck contains the EOJ (end of job) command.

Lexical Functions

A set of functions that return information about character strings and attributes of the current process.

description

The command language includes constructs, called lexical functions, that return information about the current process and about arithmetic and string expressions. The general format of a lexical function is as follows:

F$function-name([args,...])

F$	Indicates that what follows is a lexical function.
function-name	A keyword specifying the function to be evaluated. Function names can be truncated to any unique abbreviation.
()	Enclose function arguments, if any. The parentheses are required for all functions, including functions that do not accept any arguments.
args,...	Specify arguments for the function, if any, using integer or character string expressions.

Table DCL-1 lists each lexical function and briefly describes the information that each function returns. A detailed description of each function, including examples, is given in the following pages.

Table DCL-1: Summary of Lexical Functions

Function	Description
F$CVSI	Extracts bit fields from character string data and converts the result, as a signed value, to an integer.
F$CVTIME	Retrieves information about an absolute, combination, or delta time string.
F$CVUI	Extracts bit fields from character string data and converts the result, as an unsigned value, to an integer.
F$DIRECTORY	Returns the current default directory name string.
F$EDIT	Edits a character string based on the edits specified.
F$ELEMENT	Extracts an element from a string in which the elements are separated by a specified delimiter.
F$ENVIRONMENT	Obtains information about the DCL command environment.
F$EXTRACT	Extracts a substring from a character string expression.
F$FAO	Invokes the $FAO system service to convert the specified control string to a formatted ASCII output string.

Table DCL-1 (Cont.): Summary of Lexical Functions

Function	Description
F$FILE_ATTRIBUTES	Returns attribute information for a specified file.
F$GETDVI	Invokes the $GETDVI system service to return a specified item of information for a specified device.
F$GETJPI	Invokes the $GETJPI system service to return accounting, status, and identification information for a process.
F$GETQUI	Invokes the $GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the system job queue file.
F$GETSYI	Invokes the $GETSYI system service to return status and identification information about the local system, or about a node in the local cluster, if your system is part of a cluster.
F$IDENTIFIER	Converts an identifier in named format to its integer equivalent, or vice versa.
F$INTEGER	Returns the integer equivalent of the result of the specified expression.
F$LENGTH	Returns the length of a specified string.
F$LOCATE	Locates a character or character substring within a string and returns its offset within the string.
F$LOGICAL	Translates a logical name and returns the equivalence name string. (Superseded in function by F$TRNLNM.)
F$MESSAGE	Returns the message text associated with a specified system status code value.
F$MODE	Shows the mode in which a process is executing.
F$PARSE	Invokes the $PARSE RMS service to parse a file specification and return either the expanded file specification or the particular file specification field that you request.
F$PID	For each invocation, returns the next process identification number in sequence.
F$PRIVILEGE	Returns a value of "TRUE" or "FALSE" depending on whether your current process privileges match the privileges listed in the argument.
F$PROCESS	Returns the current process name string.
F$SEARCH	Invokes the $SEARCH RMS service to search a directory file, and returns the full file specification for a file you name.
F$SETPRV	Sets the specified privileges and returns a list of keywords indicating the previous state of these privileges for the current process.
F$STRING	Returns the string equivalent of the result of the specified expression.

Table DCL-1 (Cont.): Summary of Lexical Functions

Function	Description
F$TIME	Returns the current date and time of day, in the format dd-mm-yyy hh:mm:ss.cc.
F$TRNLNM	Translates a logical name and returns the equivalence name string or the requested attributes of the logical name.
F$TYPE	Determines the data type of a symbol.
F$USER	Returns the current user identification code (UIC).
F$VERIFY	Returns the integer 1 if command procedure verification is set on; returns the integer 0 if command procedure verification is set off. The F$VERIFY function also can set new verification states.

F$CVSI

Converts the specified bits in the specified character string to a signed number.

format

F$CVSI*(start-bit,number-of-bits,string)*

arguments

start-bit
The offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

number-of-bits
The length of the bit string to be extracted, which must be less than or equal to the number of bits in the string.

string
The string from which the bits are taken. Specify the string as a character string expression.

example

```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
  A = "+..."
$ X = F$CVSI(0,4,A)
$ SHOW SYMBOL X
  X = -5   Hex = FFFFFFFB  Octal = 37777777773
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. See the description of the Assignment Statement for more information on arithmetic overlays.

The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII value of the plus sign (+).

Next, the F$CVSI function extracts the low-order 4 bits from the symbol A; the low order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, –5, is assigned to the symbol X.

F$CVTIME

Converts an absolute or a combination time string to a string of the form *yyyy-mm-dd hh:mm:ss.cc*. The F$CVTIME function can also return information about an absolute, combination, or delta time string.

format

F$CVTIME*([input_time] [,output_time_format] [,output_field])*

arguments

input_time

Specifies a string containing an absolute, combination, or delta time, or TODAY, TOMORROW, or YESTERDAY. Specify the input time string as a character string expression. If the input_time argument is omitted or specified as a null string ("""), the current system date and time, in absolute format, is used. If parts of the date field are omitted, the missing values default to the current date. If parts of the time field are omitted, the missing values default to zero. If the input_time argument is a delta time, you must specify the output_time_format argument as DELTA.

output_time_format

Specifies the time format for the information you want returned. Specify the output_time_format argument as one of the following character string expressions:

ABSOLUTE	The requested information should be returned in absolute time format, which is *dd-mmm-yyyy hh:mm:ss.cc*.
COMPARISON (default)	The requested information should be returned in the form *yyyy-mm-dd hh:mm:ss.cc*; used for comparing two times.
DELTA	The requested information should be returned in delta format, which is *dddd-hh:mm:ss.cc*. If the input_time argument is a delta time, the output_time_format argument must be DELTA.

output_field

Specifies a character string expression containing one of the following (do not abbreviate): DATE, MONTH, DATETIME (default), SECOND, DAY, TIME, HOUR, WEEKDAY, HUNDREDTH, YEAR, MINUTE. The information is returned in the time format specified by the output_time_format argument. If the input_time argument is a delta time and the output_time_format argument is DELTA, you cannot specify MONTH, WEEKDAY, or YEAR.

example

```
$ TIME = F$TIME()
$ SHOW SYMBOL TIME
  TIME = "15-APR-1988 10:56:23.10"
$ TIME = F$CVTIME(TIME)
$ SHOW SYMBOL TIME
  TIME = "1988-04-15 10:56:23.10"
```

This example uses the F$TIME function to return the system time as a character string and to assign the time to the symbol TIME. Then the F$CVTIME function is used to convert the system time to an alternate time format. Note that you do not need to place quotation marks around the argument TIME because it is a symbol. Symbols are automatically evaluated when they are used as arguments for lexical functions.

You can use the resultant string to compare two dates (using .LTS. and .GTS. operators). For example, you can use F$CVTIME to convert two time strings and store the results in the symbols TIME_1 and TIME_2. You can compare the two values, and branch to a label, based on the following results:

```
$ IF TIME_1 .LTS. TIME_2 THEN GOTO FIRST
```

F$CVUI

Extracts bit fields from character string data and converts the result to an unsigned number.

format

F$CVUI*(start-bit,number-of-bits,string)*

arguments

start-bit
Specifies the offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

number-of-bits
Specifies the length of the bit-string to be extracted, which must be less than or equal to the number of bits in the string argument.

string
Specifies the character string to be edited.

example

```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
  A = "+ . . .  "
$ X = F$CVUI(0,4,A)
$ SHOW SYMBOL X
  X = 11   Hex = 0000000B  Octal = 00000000013
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII character "+".

Next, the F$CVUI function extracts the low-order 4 bits from the symbol A; the low-order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, 11, is assigned to the symbol X.

F$DIRECTORY

Returns the current default directory name string. The F$DIRECTORY function has no arguments, but must be followed by parentheses.

format

F$DIRECTORY()

arguments

None.

example

```
$ SAVE_DIR = F$DIRECTORY()
$ SET DEFAULT [MALCOLM.TESTFILES]
    .
    .
    .
$ SET DEFAULT 'SAVE_DIR'
```

This example shows an excerpt from a command procedure that uses the F$DIRECTORY function to save the current default directory setting. The assignment statement equates the symbol SAVE_DIR to the current directory. Then the SET DEFAULT command establishes a new default directory. Later, the symbol SAVE_DIR is used in the SET DEFAULT command that restores the original default directory.

Note that you can use the F$ENVIRONMENT function with the DEFAULT keyword to return the default disk and directory. You should use the F$ENVIRONMENT function rather than the F$DIRECTORY function in situations involving more than one disk.

F$EDIT

Edits the character string based on the edits specified in the edit-list.

format

F$EDIT*(string, edit-list)*

arguments

string

A character string to be edited. Quoted sections of the string are not edited.

edit-list

A character string containing one or more of the following keywords that specify the types of edits to be made to the string. If you use a list of keywords, separate them with commas. Do not abbreviate these keywords.

Edit	Action
COLLAPSE	Removes all spaces or tabs
COMPRESS	Replaces multiple spaces or tabs with a single space
LOWERCASE	Changes all uppercase characters to lowercase
TRIM	Removes leading and trailing spaces or tabs
UNCOMMENT	Removes comments
UPCASE	Changes all lowercase characters to uppercase

example

```
$ LINE = "    THIS    LINE    CONTAINS A ""  QUOTED  "" WORD"
$ SHOW SYMBOL LINE
  LINE =  "    THIS    LINE    CONTAINS A "  QUOTED  " WORD"
$ NEW_LINE = F$EDIT(LINE, "COMPRESS, TRIM")
$ SHOW SYMBOL NEW_LINE
  NEW_LINE = "THIS LINE CONTAINS A "  QUOTED  " WORD"
```

This example uses the F$EDIT function to compress and trim a string by replacing multiple blanks with a single blank, and by removing leading and trailing blanks. The string LINE contains quotation marks around the word QUOTED. (To enter quotation marks into a character string, use double quotation marks in the assignment statement.)

Note that the F$EDIT function does not compress the spaces in the quoted section of the string; therefore, the spaces are retained around the word QUOTED.

F$ELEMENT

Extracts one element from a string of elements.

format

F$ELEMENT*(element-number, delimiter, string)*

arguments

element-number

The number of the element to extract (numbering begins with zero). Specify the element-number argument as an integer expression. If the element-number argument exceeds the number of elements in the string, F$ELEMENT returns the delimiter.

delimiter

A character used to separate the elements in the string. Specify the delimiter as a character string expression.

string

A string containing a delimited list of elements. Specify the string as a character string expression

example

```
$ DAY_LIST = "MON/TUE/WED/THU/FRI/SAT/SUN"
$ INQUIRE DAY "ENTER DAY (MON TUE WED THU FRI SAT SUN)"
$ NUM = 0
$ LOOP:
$       LABEL = F$ELEMENT(NUM,"/",DAY_LIST)
$       IF LABEL .EQS. "/" THEN GOTO ERROR
$       IF DAY .EQS. LABEL THEN GOTO 'LABEL'
$       NUM = NUM +1
$       GOTO LOOP
$
$ MON:
    .
    .
    .
```

This example sets up a loop to test an input value against the elements in a list of values. If the value for DAY matches one of the elements in DAY_LIST, control is passed to the corresponding label. If the value returned by the F$ELEMENT function matches the delimiter, the value DAY was not present in the DAY_LIST, and control is passed to the label ERROR.

F$ENVIRONMENT

Returns information about the current DCL command environment.

format

F$ENVIRONMENT*(item)*

argument

item

A keyword, specified as a character string, that specifies the type of information to be returned. Do not abbreviate these keywords. Specify one of the following keywords:

Item	Data Type	Information Returned
CAPTIVE	String	TRUE if you are logged in to a captive account.
CONTROL	String	Control characters currently enabled with SET CONTROL. Multiple characters are separated by commas; if no control characters are enabled, the null string ("") is returned.

Item	Data Type	Information Returned
DEFAULT	String	Current default device and directory name. The returned string is the same as SHOW DEFAULT output.
DEPTH	Integer	Current command procedure depth.
INTERACTIVE	String	TRUE if the process is executing interactively.
KEY_STATE	String	Current locked keypad state. See the description of the DEFINE/KEY command for more information on keypad states.
MAX_DEPTH	Integer	Maximum allowable command procedure depth.
MESSAGE	String	Current setting of SET MESSAGE qualifiers. Each qualifier in the string is prefaced by a slash; therefore, the output from F$ENVIRONMENT("MESSAGE") can be appended to the SET MESSAGE command to form a valid DCL command line.
NOCONTROL	String	Control characters currently disabled with SET NOCONTROL. Multiple characters are separated by commas; if no control characters are disabled, the null string is returned.
ON_CONTROL_Y	String	If issued from a command procedure, returns TRUE if ON_CONTROL_Y is set. ON_CONTROL_Y always returns FALSE at DCL command level.
ON_SEVERITY	String	If issued from a command procedure, returns the severity level at which the action specified with the ON command is performed. ON_SEVERITY returns "NONE" when SET NOON is in effect or at DCL command level.
OUTPUT_RATE	String	Delta time string containing the default output rate, which indicates how often data is written to the batch job log file while the batch job is executing. OUTPUT_RATE returns a null string if used interactively.
PROCEDURE	String	File specification of the current command procedure. PROCEDURE returns a null string if used interactively.
PROMPT	String	Current DCL prompt.
PROMPT_CONTROL	String	TRUE if a carriage return and line feed precede the prompt.
PROTECTION	String	Current default file protection.
SYMBOL_SCOPE	String	[NO]LOCAL,[NO]GLOBAL to indicate the current symbol scoping state.

Item	Data Type	Information Returned
VERIFY_IMAGE	String	TRUE if image verification (SET VERIFY=IMAGE) is in effect.
VERIFY_PROCEDURE	String	TRUE if procedure verification SET VERIFY=PROCEDURE is in effect.

example

```
$ SAVE_MESSAGE = F$ENVIRONMENT("MESSAGE")
$ SET MESSAGE/NOFACILITY/NOIDENTIFICATION
.
.
.
$ SET MESSAGE'SAVE_MESSAGE'
```

This example uses the F$ENVIRONMENT function to save the current message setting before changing the setting. At the end of the command procedure, the original message setting is restored. The apostrophes surrounding the symbol SAVE_MESSAGE indicate that the value for the symbol should be substituted.

F$EXTRACT

Extracts the specified characters from the specified string.

format

F$EXTRACT*(start,length,string)*

arguments

start

Specifies the offset of the starting character of the string you want to extract. Specify the start argument as an integer expression that is greater than or equal to 0.

length

Specifies the number of characters you want to extract; must be less than or equal to the size of the string. Specify the length as an integer expression that is greater than or equal to 0.

string

Specifies the character string to be edited. Specify the string as a character string expression.

example

```
$ IF F$EXTRACT(12,2,F$TIME()) .GES. "12" THEN GOTO AFTERNOON
$ MORNING:
$ WRITE SYS$OUTPUT "Good morning!"
$ EXIT
$ AFTERNOON:
$ WRITE SYS$OUTPUT "Good afternoon!"
$ EXIT
```

This example shows a procedure that displays a different message, depending on whether the current time is morning or afternoon. It first obtains the current time of day by using the F$TIME function. The F$TIME function returns a character string, which is the string argument for the F$EXTRACT function. The F$TIME function is automatically evaluated when it is used as an argument, so you do not need to use quotation marks.

Next, the F$EXTRACT function extracts the hours from the date and time string returned by F$TIME. The string returned by F$TIME always contains the hours field beginning at an offset of 12 characters from the start of the string.

The F$EXTRACT function extracts two characters from the string, beginning at this offset, and compares the string value extracted with the string value 12. If the comparison is true, then the procedure writes "Good afternoon!". Otherwise, it writes "Good morning!".

Note that you can also use the F$CVTIME function to extract the hour field from a time specification. This method is easier than the one shown in the above example.

F$FAO

Invokes the $FAO system service to convert character and numeric input to character strings. (FAO stands for formatted ASCII output.) By specifying formatting instructions, you can use the F$FAO function to convert integer values to character strings, insert carriage returns and form feeds, insert text, and so on.

format

F$FAO*(control-string[,arg1,arg2...arg15])*

arguments

control-string

Specifies the fixed text of the output string, consisting of text and any number of FAO directives. The control string may be any length. Specify the control string as a character string expression.

Table DCL–2 lists the FAO directives you can specify in a control string.

arg1,arg2...arg15

Specifies the arguments required by the FAO directives used in the control string. Specify the arguments arg1,arg2...arg15 as integer or character string expressions. Table DCL–2 lists the argument types required by each FAO directive.

If you specify an argument whose type (integer or string) does not match that of the corresponding directive, unpredictable results are returned. You can use the F$INTEGER and F$STRING lexical functions to convert arguments to the proper type.

description

Specify an FAO directive using any one of the following formats:

Format	Function
!DD	One directive
!n(DD)	A directive repeated a specified number of times
!lengthDD	A directive that places its output in a field of a specified length
!n(lengthDD)	A directive that is repeated a specified number of times and generates output fields of a specified length

The exclamation point (!) indicates that the following character or characters are to be interpreted as an FAO directive. DD represents a one- or two-character uppercase code indicating the action that F$FAO is to perform. When specifying repeat counts, n is a decimal value specifying the number of times the directive is to be repeated. The length value is a decimal value that instructs F$FAO to generate an output field of "length" characters.

The FAO directives are grouped in the following categories:

- Character string insertion
- Zero-filled numeric conversion
- Blank-filled numeric conversion

- Special formatting
- Parameter interpretation

Table DCL-2 summarizes the FAO directives and shows the required argument types.

Table DCL-2: Summary of FAO Directives

Directive	Argument Type	Description
Character string insertion:		
!AS	String	Inserts a character string as is
Zero-filled numeric conversion:		
!OB	Integer	Converts a byte to octal notation
!OW	Integer	Converts a word to octal notation
!OL	Integer	Converts a longword to octal notation
!XB	Integer	Converts a byte to hexadecimal notation
!XW	Integer	Converts a word to hexadecimal notation
!XL	Integer	Converts a longword to hexadecimal notation
!ZB	Integer	Converts a byte to decimal notation
!ZW	Integer	Converts a word to decimal notation
!ZL	Integer	Converts a longword to decimal notation
Blank-filled numeric conversion:		
!UB	Integer	Converts a byte to decimal notation without adjusting for negative numbers
!UW	Integer	Converts a word to decimal notation without adjusting for negative numbers
!UL	Integer	Converts a longword to decimal notation without adjusting for negative numbers
!SB	Integer	Converts a byte to decimal notation with negative numbers converted properly
!SW	Integer	Converts a word to decimal notation with negative numbers converted properly
!SL	Integer	Converts a longword to decimal notation with negative numbers converted properly
Special formatting:		
!/	None	Inserts a carriage return and a line feed
!_	None	Inserts a tab
!^	None	Inserts a form feed
!!	None	Inserts an exclamation mark

Table DCL-2 (Cont.): Summary of FAO Directives

Directive	Argument Type	Description
!%I	Integer	Converts a longword integer to a named UIC in the format [group-identifier,member-identifier]
!%S	None	Inserts an "s" if the most recently converted number is not 1 (Not recommended for use with multilingual products.)
!%U	Integer	Converts a longword integer to a numeric UIC in the format [g,m], where g is the group number and m is the member number
		The directive inserts the brackets and the comma
!n <...!>	None	Left-justifies and blank-fills all data represented by the instructions ... in fields *n* characters wide
!n*c	None	Repeats the character represented by c for n times
!%T	Integer equal to 0	Inserts the current time
!%D	Integer equal to 0	Inserts the current date/time
Argument interpretation:		
!-	None	Reuses the last argument
!+	None	Skips the next argument

Output Strings from Character String Insertion

The !AS directive inserts a character string (specified as an argument for the directive) into the control string. The field length of the character string when it is inserted into the control string defaults to the length of the character string. If the default length is shorter than an explicitly stated field length, the string is left-justified and blank-filled. If the default length is longer than an explicitly stated field length, the string is truncated on the right.

Output Strings from Zero-Filled Numeric Conversion

Directives for zero-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal, octal, or hexadecimal notation. The ASCII representation of the integer is inserted into the control string. Default output field lengths for the converted argument are determined as follows.

Directives that convert arguments to octal notation return 3 digits for byte conversion, 6 digits for word conversion, and 11 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to hexadecimal notation return 2 digits for byte conversion, 4 digits for word conversion, and 8 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to decimal notation return the required number of characters for the decimal number. Explicit-length fields longer than the default are zero-filled on the left. If an explicit-length field is shorter than the number of characters required for the decimal number, the output field is completely filled with asterisks (*).

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

Output Strings from Blank-Filled Numeric Conversion

Directives for blank-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal notation. These directives can convert the integer as a signed or unsigned number. The ASCII representation of the integer is inserted into the control string.

Output field lengths for the converted argument default to the required number of characters. Values shorter than explicit-length fields are right-justified and blank-filled; values longer than explicit-length fields cause the field to be filled with asterisks.

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

example

```
$ COUNT = 57
$ REPORT = F$FAO("NUMBER OF FORMS = !SL",COUNT)
$ SHOW SYMBOL REPORT
$ REPORT = "NUMBER OF FORMS = 57"
```

In this command procedure, the FAO directive !SL is used in a control string to convert the number equated to the symbol COUNT to a character string. The converted string is inserted into the control string.

Note that COUNT is assigned an integer value of 57. The F$FAO function returns the ASCII string, "NUMBER OF FORMS = 57", and assigns the string to the symbol REPORT.

F$FILE_ATTRIBUTES

Returns attribute information for a specified file.

format

F$FILE_ATTRIBUTES*(file-spec,item)*

arguments

file-spec
Specifies the name of the file, as a character string, about which you are requesting information. Only one file name may be specified. Wildcard characters are not allowed.

item
Indicates which attribute of the file is to be returned. The item argument must be specified as a character string expression, and can be any one of the VMS RMS field names listed in Table DCL–3.

Table DCL–3: F$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
ALQ	Integer	Allocation Quantity
BDT	String	Backup Date/Time
BKS	Integer	Bucket Size
BLS	Integer	Block Size
CBT	String	"TRUE" If Contiguous-Best-Try; returns "TRUE" or "FALSE"
CDT	String	Creation Date/Time
CTG	String	"TRUE" If Contiguous; returns "TRUE" or "FALSE"
DEQ	Integer	Default Extension Quantity
DID	String	Directory ID String
DVI	String	Device Name String
EDT	String	Expiration Date/Time
EOF	Integer	Number of Blocks Used
FID	String	File ID String
FSZ	Integer	Fixed Control Area Size
GRP	Integer	Owner Group Number
KNOWN	String	Known File; returns "TRUE" or "FALSE" to indicate whether file is installed with the Install Utility
MBM	Integer	Owner Member Number
MRN	Integer	Maximum Record Number

Table DCL-3 (Cont.): F$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
MRS	Integer	Maximum Record Size
NOA	Integer	Number of Areas
NOK	Integer	Number of Keys
ORG	String	File Organization; returns "SEQ", "REL", "IDX"
PRO	String	File Protection String
PVN	Integer	Prolog Version Number
RAT	String	Record Attributes; returns "CR", "PRN", "FTN", ""
RCK	String	TRUE If Read Check; returns "TRUE", "FALSE"
RDT	String	Revision Date/Time
RFM	String	Record Format String; returns the values "VAR", "FIX", "VFC", "UDF", "STM", "STMLF","STMCR"
RVN	Integer	Revision Number
UIC	String	Owner UIC String
WCK	String	True If Write Check; returns "TRUE","FALSE"

example

```
$ FILE_ORG = F$FILE_ATTRIBUTES("QUEST.DAT","ORG")
$ SHOW SYMBOL FILE_ORG
  FILE_ORG = "SEQ"
```

This example uses the F$FILE_ATTRIBUTES function to assign the value of the file organization type to the symbol FILE_ORG. The F$FILE_ATTRIBUTES function returns the character string "SEQ" to show that QUEST.DAT is a sequential file.

The QUEST.DAT and ORG arguments for the F$FILE_ATTRIBUTES function are string literals and must be enclosed in quotation marks when used in expressions.

F$GETDVI

Invokes the $GETDVI system service to return a specified item of information for a specified device.

format

F$GETDVI*(device-name,item)*

arguments

device-name

Specifies a physical device name or a logical name equated to a physical device name. Specify the device name as a character string expression.

item

Specifies the type of device information to be returned. The item argument must be specified as a character string expression and may be any one of the items listed in Table DCL–4.

description

The F$GETDVI function returns information on all items that can be specified with the $GETDVI system service. In addition to the items that the $GETDVI system service allows, the F$GETDVI function allows you to specify the item EXISTS.

Table DCL–4 lists the items you can specify with the F$GETDVI function, the type of information returned, and the data types of the return values.

Table DCL–4: F$GETDVI Items

Item	Return Type	Information Returned
ACPPID	String	ACP process ID.
ACPTYPE	String	ACP type code, as one of the following strings: "F11CV1", "F11V2", "JNL", "MTA", "NET", or "REM".
ALL	String	"TRUE" or "FALSE" to indicate whether the device is allocated.
ALLDEVNAM	String	Allocation class device name.
ALLOCLASS	Longword integer between 0 and 255	Allocation class of the host.
ALT_HOST_AVAIL	String	"TRUE" or "FALSE" to indicate whether the host serving the alternate path is available.
ALT_HOST_NAME	String	Name of the host serving the alternate path.
ALT_HOST_TYPE	String	Hardware type of the host serving the alternate path.
AVL	String	"TRUE" or "FALSE" to indicate whether the device is available for use.
CCL	String	"TRUE" or "FALSE" to indicate whether the device is a carriage control device.
CLUSTER	Integer	Volume cluster size.

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
CONCEALED	String	"TRUE" or "FALSE" to indicate whether the logical device name translates to a concealed device.
CYLINDERS	Integer	Number of cylinders on the volume (disk).
DEVBUFSIZ	Integer	Device buffer size.
DEVCHAR	Integer	Device characteristics.
DEVCHAR2	Integer	Additional device characteristics.
DEVCLASS	Integer	Device class. See Table DCL-5 for a list of the values returned.
DEVDEPEND	Integer	Device-dependent information.
DEVDEPEND2	Integer	Additional device-dependent information.
DEVLOCKNAM	String	A unique lock name for the device.
DEVNAM	String	Device name.
DEVSTS	Integer	Device-dependent status information.
DEVTYPE	Integer	Device type. See Table DCL-6 for a list of the values returned.
DIR	String	"TRUE" or "FALSE" to indicate whether the device is directory structured.
DMT	String	"TRUE" or "FALSE" to indicate whether the device is marked for dismount.
DUA	String	"TRUE" or "FALSE" to indicate whether the device is a generic device.
ELG	String	"TRUE" or "FALSE" to indicate whether the device has error logging enabled.
ERRCNT	Integer	Error count.
EXISTS	String	"TRUE" or "FALSE" to indicate whether the device exists on the system.
FOD	String	"TRUE" or "FALSE" to indicate whether the device is a files-oriented device.
FOR	String	"TRUE" or "FALSE" to indicate whether the device is mounted foreign.
FREEBLOCKS	Integer	Number of free blocks on the volume (disk).
FULLDEVNAM	String	Fully qualified device name.
GEN	String	"TRUE" or "FALSE" to indicate whether the device is a generic device.
HOST_AVAIL	String	"TRUE" or "FALSE" to indicate whether the host serving the primary path is available.

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
HOST_COUNT	Integer	Number of hosts that make the device available to other nodes in the VAXcluster.
HOST_NAME	String	Name of the host serving the primary path.
HOST_TYPE	String	Hardware type of the host serving the primary path.
IDV	String	"TRUE" or "FALSE" to indicate whether the device is capable of providing input.
LOCKID	Integer	Clusterwide lock identification.
LOGVOLNAM	String	Logical volume name.
MAXBLOCK	Integer	Number of logical blocks on the volume.
MAXFILES	Integer	Maximum number of files on the volume. This item code is applicable only to disks.
MBX	String	"TRUE" or "FALSE" to indicate whether the device is a mailbox.
MEDIA_ID	String	Non-decoded media ID.
MEDIA_NAME	String	Either the name of the disk or the tape type.
MEDIA_TYPE	String	Device name prefix.
MNT	String	"TRUE" or "FALSE" to indicate whether the device is mounted.
MOUNTCNT	Integer	Mount count.
NET	String	"TRUE" or "FALSE" to indicate whether the device is a network device.
NEXTDEVNAM	String	Device name of the next volume in a volume set. This item applies only to disks.
ODV	String	"TRUE" or "FALSE" to indicate whether the device is capable of providing output.
OPCNT	Integer	Operation count.
OPR	String	"TRUE" or "FALSE" to indicate whether the device is an operator.
OWNUIC	String	UIC of the device owner.
PID	String	Process identification of the device owner.
RCK	String	"TRUE" or "FALSE" to indicate whether the device has read checking enabled.
RCT	String	"TRUE" or "FALSE" to indicate whether the disk contains RCT.
REC	String	"TRUE" or "FALSE" to indicate whether the device is record oriented.
RECSIZ	Integer	Blocked record size.

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
REFCNT	Integer	Reference count of processes using the device.
REMOTE_DEVICE	String	"TRUE" or "FALSE" to indicate whether the device is a remote device.
RND	String	"TRUE" or "FALSE" to indicate whether the device allows random access.
ROOTDEVNAM	String	Device name of the root volume in a volume set. This item applies only to disks.
RTM	String	"TRUE" or "FALSE" to indicate whether the device is real-time.
SDI	String	"TRUE" or "FALSE" to indicate whether the device is single-directory structured.
SECTORS	Integer	Number of sectors per track. This item applies only to disks.
SERIALNUM	Integer	Volume serial number. This item applies only to disks.
SERVED_DEVICE	String	"TRUE" or "FALSE" to indicate whether the device is a served device.
SHR	String	"TRUE" or "FALSE" to indicate whether the device is shareable.
SPL	String	"TRUE" or "FALSE" to indicate whether the device is being spooled.
SPLDEVNAM	String	Name of the device being spooled.
SQD	String	"TRUE" or "FALSE" to indicate whether the device is sequential block-oriented (that is, magnetic tape).
STS	Integer	Status information.
SWL	String	"TRUE" or "FALSE" to indicate whether the device is software write-locked.
TRACKS	Integer	Number of tracks per cylinder. This item applies only to disks.
TRANSCNT	Integer	Volume transaction count.
TRM	String	"TRUE" or "FALSE" to indicate whether the device is a terminal.
TT_ACCPORNAM	String	The terminal server name and port name.
TT_ALTYPEAHD	String	"TRUE" or "FALSE" to indicate whether the terminal has an alternate type-ahead buffer (terminals only).
TT_ANSICRT	String	"TRUE" or "FALSE" to indicate whether the terminal is an ANSI CRT terminal (terminals only).
TT_APP_KEYPAD	String	"TRUE" or "FALSE" to indicate whether the keypad is in applications mode (terminals only).

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
TT_AUTOBAUD	String	"TRUE" or "FALSE" to indicate whether the terminal has automatic baud rate detection (terminals only).
TT_AVO	String	"TRUE" or "FALSE" to indicate whether the terminal has a VT100-family terminal display (terminals only).
TT_BLOCK	String	"TRUE" or "FALSE" to indicate whether the terminal has block mode capability (terminals only).
TT_BRDCSTMBX	String	"TRUE" or "FALSE" to indicate whether the terminal uses mailbox broadcast messages (terminals only).
TT_CRFILL	String	"TRUE" or "FALSE" to indicate whether the terminal requires fill after RET (terminals only).
TT_DECCRT	String	"TRUE" or "FALSE" to indicate whether the terminal is a DIGITAL CRT terminal (terminals only).
TT_DECCRT2	String	"TRUE" or "FALSE" to indicate whether the terminal is a DIGITAL CRT2 terminal (terminals only).
TT_DIALUP	String	"TRUE" or "FALSE" to indicate whether the terminal is connected to dialup (terminals only).
TT_DISCONNECT	String	"TRUE" or "FALSE" to indicate whether the terminal can be disconnected (terminals only).
TT_DMA	String	"TRUE" or "FALSE" to indicate whether the terminal has DMA mode (terminals only).
TT_DRCS	String	"TRUE" or "FALSE" to indicate whether the terminal supports loadable character fonts (terminals only).
TT_EDIT	String	"TRUE" or "FALSE" to indicate whether the edit characteristic is set.
TT_EDITING	String	"TRUE" or "FALSE" to indicate whether advanced editing is enabled (terminals only).
TT_EIGHTBIT	String	"TRUE" or "FALSE" to indicate whether the terminal uses the 8-bit ASCII character set (terminals only).
TT_ESCAPE	String	"TRUE" or "FALSE" to indicate whether the terminal generates escape sequences (terminals only).
TT_FALLBACK	String	"TRUE" or "FALSE" to indicate whether the terminal uses the multinational fallback option (terminals only).
TT_HALFDUP	String	"TRUE" or "FALSE" to indicate whether the terminal is in half-duplex mode (terminals only).
TT_HANGUP	String	"TRUE" or "FALSE" to indicate whether the hangup characteristic is set (terminals only).
TT_HOSTSYNC	String	"TRUE" or "FALSE" to indicate whether the terminal has host/terminal communication (terminals only).
TT_INSERT	String	"TRUE" or "FALSE" to indicate whether insert-mode is the default line editing mode (terminals only).

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
TT_LFFILL	String	"TRUE" or "FALSE" to indicate whether the terminal requires fill after LF (terminals only).
TT_LOCALECHO	String	"TRUE" or "FALSE" to indicate whether the local echo characteristic is set (terminals only).
TT_LOWER	String	"TRUE" or "FALSE" to indicate whether the terminal has the lowercase characters set.
TT_MBXDSABL	String	"TRUE" or "FALSE" to indicate whether mailboxes associated with the terminal will receive unsolicited input notification or input notification (terminals only).
TT_MECHFORM	String	"TRUE" or "FALSE" to indicate whether the terminal has mechanical form feed (terminals only).
TT_MECHTAB	String	"TRUE" or "FALSE" to indicate whether the terminal has mechanical tabs and is capable of tab expansion (terminals only).
TT_MODEM	String	"TRUE" or "FALSE" to indicate whether the terminal is connected to a modem (terminals only).
TT_MODHANGUP	String	"TRUE" or "FALSE" to indicate whether the modify hang-up characteristic is set (terminals only).
TT_NOBRDCST	String	"TRUE" or "FALSE" to indicate whether the terminal will receive broadcast messages (terminals only).
TT_NOECHO	String	"TRUE" or "FALSE" to indicate whether the input characters are echoed.
TT_NOTYPEAHD	String	"TRUE" or "FALSE" to indicate whether data must be solicited by a read operation.
TT_OPER	String	"TRUE" or "FALSE" to indicate whether the terminal is an operator terminal (terminals only).
TT_PAGE	Integer	Terminal page length (terminals only).
TT_PASTHRU	String	"TRUE" or "FALSE" to indicate whether there is passall with flow control (terminals only).
TT_PHYDEVNAM	String	Physical device name associated with a channel number or virtual terminal.
TT_PRINTER	String	"TRUE" or "FALSE" to indicate whether there is a printer port available (terminals only).
TT_READSYNC	String	"TRUE" or "FALSE" to indicate whether the terminal has read synchronization (terminals only).
TT_REGIS	String	"TRUE" or "FALSE" to indicate whether the terminal has REGIS graphics (terminals only).
TT_REMOTE	String	"TRUE" or "FALSE" to indicate whether the terminal has established modem control (terminals only).

Table DCL-4 (Cont.): F$GETDVI Items

Item	Return Type	Information Returned
TT_SCOPE	String	"TRUE" or "FALSE" to indicate whether the terminal is a video screen display (terminals only).
TT_SECURE	String	"TRUE" or "FALSE" to indicate whether the terminal can recognize the secure server (terminals only).
TT_SETSPEED	String	"TRUE" or "FALSE" to indicate whether you can set the speed on the terminal line (terminals only).
TT_SIXEL	String	"TRUE" or "FALSE" to indicate whether the sixel is supported (terminals only).
TT_TTSYNC	String	"TRUE" or "FALSE" to indicate whether there is terminal /host synchronization (terminals only).
TT_SYSPWD	String	"TRUE" or "FALSE" to indicate whether the system password is enabled for a particular terminal.
TT_WRAP	String	"TRUE" or "FALSE" to indicate whether a new line should be inserted if the cursor moves beyond the right margin.
UNIT	Integer	The unit number.
VOLCOUNT	Integer	The count of volumes in a volume set. This item applies only to disks.
VOLNAM	String	The volume name.
VOLNUMBER	Integer	Number of the current volume in a volume set. This item applies only to disks.
VOLSETMEM	String	"TRUE" or "FALSE" to indicate whether the device is a volume set (disks only).
VPRO	String	The volume protection mask.
WCK	String	"TRUE" or "FALSE" to indicate whether the device has write checking enabled.

Table DCL-5: Values Returned by the DEVCLASS Item

Device Class	Value	Symbolic Name
Disk device	1	DC$_DISK
Tape device	2	DC$_TAPE
Synchronous Communications device	32	DC$_SCOM
Card reader	65	DC$_CARD
Terminal	66	DC$_TERM
Line printer	67	DC$_LP
Real-time	96	DC$_REALTIME
Bus	128	DC$_BUS
Mailbox	160	DC$_ MAILBOX
Journal	161	DC$_JOURNAL
Miscellaneous device	200	DC$_MISC

Table DCL-6: Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
RK06	1	VT102	98
RK07	2	VT105	99
RP04	3	VT125	100
RP05	4	VT131	101
RP06	5	VT132	102
RMO3	6	DZ11	66
RP07	7	DZ32	67
RP07HT	8	DZ730	68
RL01	9	DMC11	1
RL02	10	DMR11	2
RX02	11	XK_3271	3
RX04	12	XJ_2780	4
RX33	36	NW_X25	5
CRX50	33	NV_X29	6
RM80	13	SB_ISB11	7
TU58	14	MX_MUX200	8
RM05	15	DMP11	9

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
RX01	16	DMF32	10
ML11	17	XV_3271	11
RB02	18	CI	12
RB80	19	NI	13
RA80	20	UNA11	14
RA81	21	YN_X25	15
RA60	224	YO_X25	16
RZ01	23	YP_ADCCP	17
RZF01	2	YQ_3271	18
RD51	25	YR_DDCMP	19
RX50	26	YS_SDLC	20
TE16	1	LP11	1
TK50	10	LA11	2
TK60	17	LA180	3
TK70	15	CR11	1
TU45	2	MBX	1
TU77	3	SHRMBX	2
TS11	4	NULL	3
TU78	5	LPA11	1
TA78	6	DR780	2
TU80	7	DR750	3
TU81	8	DR11W	4
TA81	9	PCL11R	5
TTYUNKN	0	PCL11T	6
VT105	1	DR11C	7
FT1	16	XI_DR11C	8
FT2	17	XP_PCL11B	9
FT3	18	CI780	1
FT4	19	C1750	2
FT5	20	UQPORT	3
FT6	21	UDA50	3
FT7	22	UDA50A	4
FT8	23	RC25	23

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
LAX	32	TU81P	6
LA36	32	RDRX	7
LA120	33	UNKNJNL	0
VT5X	64	RUJNL	1
VT52	64	BIJNL	2
VT55	65	AIJNL	3
VT100	96	ATJNL	4
VT101	97	CLJNL	5
VK100	2	DN11	1
VT173	3	VT200	110
LA34	34	LA210	40
LA38	35	LA100	37
LA12	36	LQP02	38
LA24	37		

example

```
$ ERR = F$GETDVI("_DQA0","ERRCNT")
$ SHOW SYMBOL ERR
  ERR = 0  Hex = 00000000 Octal = 000000
```

This example shows how to use the F$GETDVI function to return an error count for the device DQA0. You must place quotation marks around the device name DQA0 and the item ERRCNT because they are string literals.

F$GETJPI

Invokes the $GETJPI system service to return accounting, status, and identification information on the specified process.

Requires GROUP privilege to obtain information on other processes in the same group. Requires WORLD privilege to obtain information on any other processes in the system.

format

F$GETJPI*(pid,item)*

arguments

pid

Specifies the identification number of the process for which information is being reported. Specify the pid argument as a character string expression. You can omit the leading zeros. If you specify a null string ("), the current process identification number is used.

item

Indicates the type of process information to be returned. *Item* must be specified as a character string expression and may be any one of the items listed in Table DCL–7.

description

The F$GETJPI function returns information on all items that can be specified with the $GETJPI system service.

Table DCL–7: F$GETJPI Items

Item	Return Type	Information Returned
ACCOUNT	String	Account name string (8 characters filled with trailing blanks)
APTCNT	Integer	Active page table count
ASTACT	Integer	Access modes with active ASTs
ASTCNT	Integer	Remaining AST quota
ASTEN	Integer	Access modes with ASTs enabled
ASTLM	Integer	AST limit quota
AUTHPR	Integer	Maximum priority that a process without the ALTPRI privilege can achieve with the $SETPRI system service
AUTHPRIV	String	Privileges that a process is authorized to enable
BIOCNT	Integer	Remaining buffered I/O quota
BIOLM	Integer	Buffered I/O limit quota
BUFIO	Integer	Count of process buffered I/O operations
BYTCNT	Integer	Remaining buffered I/O byte count quota
BYTLM	Integer	Buffered I/O byte count limit quota
CLINAME	String	Current command language interpreter; always returns "DCL"
CPULIM	Integer	Limit on process CPU time
CPUTIM	Integer	CPU time used in hundredths of a second
CURPRIV	String	Current process privileges

Table DCL-7 (Cont.): F$GETJPI Items

Item	Return Type	Information Returned
DFPFC	Integer	Default page fault cluster size
DFWSCNT	Integer	Default working set size
DIOCNT	Integer	Remaining direct I/O quota
DIOLM	Integer	Direct I/O limit quota
DIRIO	Integer	Count of direct I/O operations for the process
EFCS	Integer	Local event flags 0 through 31
EFCU	Integer	Local event flags 32 through 63
EFWM	Integer	Event flag wait mask
ENQCNT	Integer	Lock request quota remaining
ENQLM	Integer	Lock request quota limit
EXCVEC	Integer	Address of a list of exception vectors
FILCNT	Integer	Remaining open file quota
FILLM	Integer	Open file quota
FINALEXC	Integer	Address of a list of final exception vectors
FREP0VA	Integer	First free page at end of program region (PO space)
FREP1VA	Integer	First free page at end of control region (P1 space)
FREPTECNT	Integer	Number of pages available for virtual memory expansion
GPGCNT	Integer	Global page count in working set
GRP	Integer	Group number of the UIC
IMAGECOUNT	Integer	Number of images that have been run down for the process
IMAGNAME	String	File name of the current image
IMAGPRIV	String	Privileges with which the current image was installed
JOBPRCCNT	Integer	Number of subprocesses owned by the process
LOGINTIM	String	Process creation time
MASTER_PID	String	Returns the process identification of the process at the top of the current job's process tree
MEM	Integer	Member number of the UIC
MODE	String	Current process mode ("BATCH", "INTERACTIVE", "NETWORK", or "OTHER")
MSGMASK	Integer	Default message mask
OWNER	String	Process identification number of process owner
PAGEFLTS	Integer	Count of page faults
PAGFILCNT	Integer	Remaining paging file quota

Table DCL-7 (Cont.): F$GETJPI Items

Item	Return Type	Information Returned
PAGFILLOC	Integer	Location of the paging file
PGFLQUOTA	Integer	Paging file quota (maximum virtual page count)
PHDFLAGS	Integer	Flags word
PID	String	Process identification number
PPGCNT	Integer	Process page count
PRCCNT	Integer	Count of subprocesses
PRCLM	Integer	Subprocess quota
PRCNAM	String	Process name
PRIB	Integer	Process's base priority
PROCPRIV	Integer	Process's default privileges
SITESPEC	Integer	Per-process site-specific longword
STATE	String	Process state
STS	Integer	Process status flags
SWPFILLOC	Integer	Location of the swap file
TERMINAL	String	Login terminal name for interactive users (1-7 characters)
TMBU	Integer	Termination mailbox unit number
TQCNT	Integer	Remaining timer queue entry quota
TQLM	Integer	Timer queue entry quota
UIC	String	Process's UIC
USERNAME	String	User name string (12 characters filled with trailing blanks)
VIRTPEAK	Integer	Peak virtual address size
VOLUMES	Integer	Count of currently mounted volumes
WSAUTH	Integer	Maximum authorized working set size
WSAUTHEXT	Integer	Maximum authorized working set extent
WSEXTENT	Integer	Current working set extent
WSPEAK	Integer	Working set peak
WSQUOTA	Integer	Working set size quota
WSSIZE	Integer	Process's current working set size

example

```
$ NAME = F$GETJPI("3B0018","USERNAME")
$ SHOW SYMBOL NAME
  NAME = "JANE        "
```

This example shows how to use the F$GETJPI function to return the username for the process number 3B0018. The username is assigned to the symbol NAME.

F$GETQUI

Invokes the $GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the system job queue file.

Requires READ access to the job or SYSPRV or OPER privilege to obtain job and file information.

format

F$GETQUI*(function,[item],[object-id],[flags])*

arguments

function

Specifies the action that the F$GETQUI lexical function is to perform. F$GETQUI supports all functions that can be specified with the $GETQUI system service.

Function	Description
CANCEL_OPERATION	Terminates any wildcard operation that may have been initiated by a previous call to F$GETQUI.
DISPLAY_CHARACTERISTIC	Returns information about a specific characteristic definition or the next characteristic definition in a wildcard operation.
DISPLAY_ENTRY	Returns information about a specific job entry or the next job entry that matches the selection criteria in a wildcard operation. The DISPLAY_ENTRY function code is similar to the DISPLAY_JOB function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established.
DISPLAY_FILE	Returns information about the next file defined for the current job context. Before you make a call to F$GETQUI to request file information, you must make a call to display queue and job information (with the DISPLAY_QUEUE and DISPLAY_JOB function codes) or display entry information (with the DISPLAY_ENTRY function code).

Function	Description
DISPLAY_FORM	Returns information about a specific form definition or the next form definition in a wildcard operation.
DISPLAY_JOB	Returns information about the next job defined for the current queue context. Before you make a call to F$GETQUI to request job information, you must make a call to display queue information (with the DISPLAY_QUEUE function code). The DISPLAY_JOB function code is similar to the DISPLAY_ENTRY function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established.
DISPLAY_QUEUE	Returns information about a specific queue definition or the next queue definition in a wildcard operation.
TRANSLATE_QUEUE	Translates a logical name for a queue to the equivalence name for the queue.

Some function arguments cannot be specified with the item-code, object-id, or flags argument. The following table lists each function argument and corresponding format line to show whether the item-code, object-id, and flags arguments are required, optional, or not applicable for that specific function. In the following format lines, brackets denote an optional argument. An omitted argument means the argument is not applicable for that function. Note that two commas must be used as placeholders to denote an omitted (whether optional or not applicable) argument.

Function	Format Line
CANCEL_OPERATION	F$GETQUI("CANCEL_OPERATION") or F$GETQUI("")
DISPLAY_CHARACTERISTIC	F$GETQUI("DISPLAY_CHARACTERISTIC",[item],object-id,[flags])
DISPLAY_ENTRY	F$GETQUI("DISPLAY_ENTRY",[item],[object-id],[flags])
DISPLAY_FILE	F$GETQUI("DISPLAY_FILE",[item],,[flags])
DISPLAY_FORM	F$GETQUI("DISPLAY_FORM",[item],object-id,[flags])
DISPLAY_JOB	F$GETQUI("DISPLAY_JOB",[item],,[flags])
DISPLAY_QUEUE	F$GETQUI("DISPLAY_QUEUE",[item],object-id,[flags])
TRANSLATE_QUEUE	F$GETQUI("TRANSLATE_QUEUE",[item],object-id)

item

Corresponds to a $GETQUI system service output item code. *Item* specifies the nature of the information you want returned about a particular queue, job, file, form, or characteristic. Table DCL–8 lists each item code and the data type of the value returned for each item code.

object-id

Corresponds to the $GETQUI system service search-name and search-number input item codes. *Object-id* specifies either the name or number of an object (for example, a specific queue name or form number) about which F$GETQUI is to return information. Wildcard names are allowed for the following functions:

- DISPLAY_CHARACTERISTIC
- DISPLAY_ENTRY
- DISPLAY_FORM
- DISPLAY_QUEUE

By specifying a wildcard as the object-id argument on successive calls, you can get status information about one or more jobs in a specific queue or about files within jobs in a specific queue. When a wildcard name is used, each call returns information for the next object (queue, form, and so on) in the list. A null string ("") is returned when the end of the list is reached. A wildcard can represent only object names, not object numbers.

flags

Specifies a list of keywords, separated by commas, that corresponds to the flags defined for the $GETQUI system service search-flags input item code. (These flags are used to define the scope of the object search specified in the call to the $GETQUI system service.) Note that these keywords can be used only with certain function codes:

Keyword	Valid Function Code	Description
ALL_JOBS	DISPLAY_JOB	Requests that F$GETQUI search all jobs included in the established queue context. If you do specify this flag, F$GETQUI returns information only about jobs that have the same user name as the caller.
BATCH	DISPLAY_QUEUE	Selects batch queues.
	DISPLAY_ENTRY	
EXECUTING_JOBS	DISPLAY_ENTRY	Selects executing jobs.
	DISPLAY_JOB	
FREEZE_CONTEXT	DISPLAY_CHARACTERISTIC	When in wildcard mode, prevents advance of wildcard context to the next object.
	DISPLAY_ENTRY	
	DISPLAY_FILE	
	DISPLAY_FORM	
	DISPLAY_JOB	
	DISPLAY_QUEUE	
GENERIC	DISPLAY_QUEUE	Selects generic queues for searching.
HOLDING_JOBS	DISPLAY_ENTRY	Selects jobs on unconditional hold.
	DISPLAY_JOB	
PENDING_JOBS	DISPLAY_ENTRY	Selects pending jobs.
	DISPLAY_JOB	
PRINTER	DISPLAY_QUEUE	Selects printer queues.
	DISPLAY_ENTRY	
RETAINED_JOBS	DISPLAY_ENTRY	Selects jobs being retained.
	DISPLAY_JOB	
SERVER	DISPLAY_QUEUE	Selects server queues.
	DISPLAY_ENTRY	
SYMBIONT	DISPLAY_QUEUE	Selects all output queues. Equivalent to specifying "PRINTER,SERVER,TERMINAL".
	DISPLAY_ENTRY	
TERMINAL	DISPLAY_QUEUE	Selects terminal queues.
	DISPLAY_ENTRY	

Keyword	Valid Function Code	Description
THIS_JOB	DISPLAY_FILE	Selects all job file information about the calling batch job, the command file being executed, or the queue associated with the calling batch job.
	DISPLAY_JOB	
	DISPLAY_QUEUE	
TIMED_RELEASE_JOBS	DISPLAY_ENTRY	Selects jobs on hold until a specified time.
	DISPLAY_JOB	
WILDCARD	DISPLAY_CHARACTERISTIC	Establishes and saves a context. Because the context is saved, the next operation can be performed based on that context.
	DISPLAY_ENTRY	
	DISPLAY_FORM	
	DISPLAY_QUEUE	

description

The F$GETQUI lexical function provides all the features of the $GETQUI system service, including wildcard and nested wildcard operations.

The F$GETQUI function returns information on all items that can be specified with the $GETQUI system service. Table DCL-8 lists the items you can specify with the F$GETQUI function, the information returned, and the data type of this information.

Table DCL-8: F$GETQUI Items

Item	Return Type	Information Returned
ACCOUNT_NAME	String	The account name of the owner of the specified job.
AFTER_TIME	String	The system time at or after which the specified job can execute.
ASSIGNED_QUEUE_NAME	String	The name of the execution queue to which the logical queue specified in the call to F$GETQUI is assigned.
BASE_PRIORITY	Integer	The priority at which batch jobs are initiated from a batch execution queue or the priority of a symbiont process that controls output execution queues.
CHARACTERISTICS	String	The characteristics associated with the specified queue or job.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
CHARACTERISTIC_NAME	String	The name of the specified characteristic.
CHARACTERISTIC_NUMBER	Integer	The number of the specified characteristic.
CHECKPOINT_DATA	String	The value of the DCL symbol BATCH$RESTART when the specified batch job is restarted.
CLI	String	The name of the command language interpreter used to execute the specified batch job. The file specification returned assumes the device name SYS$SYSTEM and the file type EXE.
COMPLETED_BLOCKS	Integer	The number of blocks that the symbiont has processed for the specified print job. This item code is applicable only to print jobs.
CONDITION_VECTOR	Integer	The completion status of the specified job.
CPU_DEFAULT	String	The default CPU time limit specified for the queue in 10-millisecond units. This item code is applicable only to batch execution queues.
CPU_LIMIT	String	The maximum CPU time limit specified for the specified job or queue in 10-millisecond units. This item code is applicable only to batch jobs and batch execution queues.
DEFAULT_FORM_NAME	String	The name of the default form associated with the specified output queue.
DEFAULT_FORM_STOCK	String	The name of the paper stock on which the specified default form is to be printed.
DEVICE_NAME	String	The node and device (or both) on which the specified execution queue is located. For batch execution queues, only the node name is returned. For output execution queues, only the device name is returned. The node name is used only in VAXcluster systems. The node name is specified by the SYSGEN parameter SCSNODE for the processor on which the queue executes.
ENTRY_NUMBER	Integer	The queue entry number of the specified job.
EXECUTING_JOB_COUNT	Integer	The number of jobs in the queue that are currently executing.
FILE_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages are to be printed preceding a file.
FILE_CHECKPOINTED	String	"TRUE" or "FALSE" to indicate whether file is checkpointed.
FILE_COPIES	Integer	The number of times the specified file is to be processed. This item code is applicable only to output execution queues.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
FILE_COPIES_DONE	Integer	The number of times the specified file has been processed. This item code is applicable only to output execution queues.
FILE_DELETE	String	"TRUE" or "FALSE" to indicate whether file is to be deleted after execution of request.
FILE_DOUBLE_SPACE	String	"TRUE" or "FALSE" to indicate whether the symbiont formats the file with double spacing.
FILE_EXECUTING	String	"TRUE" or "FALSE" to indicate whether file is being processed.
FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether a flag page is to be printed preceding a file.
FILE_FLAGS	Integer	The processing options that have been selected for the specified file.
FILE_IDENTIFICATION	String	The internal file-identification value that uniquely identifies the selected file.
FILE_PAGE_HEADER	String	"TRUE" or "FALSE" to indicate whether page header is to be printed on each page of output.
FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.
FILE_PASSALL	String	"TRUE" or "FALSE" to indicate whether symbiont prints the file in PASSALL mode.
FILE_SETUP_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before the specified file is printed. This item code is meaningful only for output execution queues.
FILE_SPECIFICATION	String	The fully qualified RMS file specification of the file about which F$GETQUI is returning information.
FILE_STATUS	Integer	File status information.
FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page is to be printed following a file.
FIRST_PAGE	Integer	The page number at which the printing of the specified file is to begin. This item code is applicable only to output execution queues.
FORM_DESCRIPTION	String	The text string that describes the specified form to users and operators.
FORM_FLAGS	Integer	The processing options that have been selected for the specified form.
FORM_LENGTH	Integer	The physical length of the specified form in lines. This item code is applicable only to output execution queues.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
FORM_MARGIN_BOTTOM	Integer	The bottom margin of the specified form in lines.
FORM_MARGIN_LEFT	Integer	The left margin of the specified form in characters.
FORM_MARGIN_RIGHT	Integer	The right margin of the specified form in characters.
FORM_MARGIN_TOP	Integer	The top margin of the specified form in lines.
FORM_NAME	String	The name of the specified form or the mounted form associated with the specified job or queue.
FORM_NUMBER	Integer	The number of the specified form.
FORM_SETUP_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before a file is printed on the specified form. This item code is meaningful only for output execution queues.
FORM_SHEET_FEED	String	"TRUE" or "FALSE" to indicate whether symbiont pauses at the end of each physical page so that another sheet of paper can be inserted.
FORM_STOCK	String	The name of the paper stock on which the specified form is to be printed.
FORM_TRUNCATE	String	"TRUE" or "FALSE" to indicate whether printer discards any characters that exceed the specified right margin.
FORM_WIDTH	Integer	The width of the specified form.
FORM_WRAP	String	"TRUE" or "FALSE" to indicate whether the printer prints any characters that exceed the specified right margin on the following line.
GENERIC_TARGET	String	The names of the execution queues that are enabled to accept work from the specified generic queue. This item code is meaningful only for generic queues.
HOLDING_JOB_COUNT	Integer	The number of jobs in the queue being held until explicitly released.
INTERVENING_BLOCKS	Integer	The number of blocks to be processed before the specified job can begin to execute. This item code is meaningful only for output execution queues.
INTERVENING_JOBS	Integer	The number of jobs that are to be processed before the specified job can begin to execute. This item code is meaningful only for output execution queues.
JOB_ABORTING	String	"TRUE" or "FALSE" to indicate whether system is attempting to abort execution of job.
JOB_COPIES	Integer	The number of times the specified print job is to be repeated.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
JOB_COPIES_DONE	Integer	The number of times that the specified print job has been repeated.
JOB_CPU_LIMIT	String	"TRUE" or "FALSE" to indicate whether a CPU time limit is specified for the job.
JOB_EXECUTING	String	"TRUE" or "FALSE" to indicate whether job is executing or printing.
JOB_FILE_BURST	String	"TRUE" or "FALSE" to indicate whether a burst page option is explicitly specified for the job.
JOB_FILE_BURST_ONE	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede only the first copy of the first file in the job.
JOB_FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether flag page precedes each file in the job.
JOB_FILE_FLAG_ONE	String	"TRUE" or "FALSE" to indicate whether flag page precedes only the first copy of the first file in the job.
JOB_FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether a paginate option is explicitly specified for the job.
JOB_FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page follows each file in the job.
JOB_FILE_TRAILER_ONE	String	"TRUE" or "FALSE" to indicate whether trailer page follows only the last copy of the last file in the job.
JOB_FLAGS	Integer	The processing options that have been selected for the specified job.
JOB_HOLDING	String	"TRUE" or "FALSE" to indicate whether job will be held until it is explicitly released.
JOB_INACCESSIBLE	String	"TRUE" or "FALSE" to indicate whether caller does not have READ access to the specific job and file information in the system queue file. Therefore, the DISPLAY_JOB and DISPLAY_FILE operations can return information for only the following output value item codes: AFTER_TIME COMPLETED_BLOCKS ENTRY_NUMBER INTERVENING_BLOCKS INTERVENING_JOBS JOB_SIZE JOB_STATUS
JOB_LIMIT	Integer	The number of jobs that can execute simultaneously on the specified queue. This item code is applicable only to batch execution queues.
JOB_LOG_DELETE	String	"TRUE" or "FALSE" to indicate whether the log file is deleted after it is printed.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
JOB_LOG_NULL	String	"TRUE" or "FALSE" to indicate whether no log file is created.
JOB_LOG_SPOOL	String	"TRUE" or "FALSE" to indicate whether job log file is queued for printing when job is complete.
JOB_LOWERCASE	String	"TRUE" or "FALSE" to indicate whether job is to be printed on printer that can print both uppercase and lowercase letters.
JOB_NAME	String	The name of the specified job.
JOB_NOTIFY	String	"TRUE" or "FALSE" to indicate whether message is broadcast to terminal when job completes or aborts.
JOB_PENDING	String	"TRUE" or "FALSE" to indicate whether job is pending.
JOB_PID	String	The process identification (PID) of the executing batch job.
JOB_REFUSED	String	"TRUE" or "FALSE" to indicate whether job was refused by symbiont and is waiting for symbiont to accept it for processing.
JOB_RESET_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before each job in the specified queue is printed. This item code is meaningful only for output execution queues.
JOB_RESTART	String	"TRUE" or "FALSE" to indicate whether job will restart after a system failure or can be requeued during execution.
JOB_RETAINED	String	"TRUE" or "FALSE" to indicate whether job has completed, but is being retained in the queue.
JOB_SIZE	Integer	The total number of blocks in the specified print job.
JOB_SIZE_MAXIMUM	Integer	The maximum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.
JOB_SIZE_MINIMUM	Integer	The minimum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.
JOB_STARTING	String	"TRUE" or "FALSE" to indicate whether job controller is starting to process the job and has begun communicating with an output symbiont or a job controller on another node.
JOB_STATUS	Integer	The specified job's status flags.
JOB_SUSPENDED	String	"TRUE" or "FALSE" to indicate whether job is suspended.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
JOB_TIMED_RELEASE	String	"TRUE" or "FALSE" to indicate whether job is waiting for specified time to execute.
JOB_WSDEFAULT	String	"TRUE" or "FALSE" to indicate whether default working set size is specified for the job.
JOB_WSEXTENT	String	"TRUE" or "FALSE" to indicate whether working set extent is specified for the job.
JOB_WSQUOTA	String	"TRUE" or "FALSE" to indicate whether working set quota is specified for the job.
LAST_PAGE	Integer	The page number at which the printing of the specified file should end. This item code is applicable only to output execution queues.
LIBRARY_SPECIFICATION	String	The name of the device control library for the specified queue. The library specification assumes the device and directory name SYS$LIBRARY and a file type of TLB. This item code is meaningful only for output execution queues.
LOG_QUEUE	String	The name of the queue into which the log file produced for the specified batch job is to be entered for printing. This item code is applicable only to batch jobs.
LOG_SPECIFICATION	String	The name of the log file to be produced for the specified job. This item code is meaningful only for batch jobs.
NOTE	String	The note that is to be printed on the job flag and file flag pages of the specified job. This item code is meaningful only for output execution queues.
OPERATOR_REQUEST	String	The message that is to be sent to the queue operator before the specified job begins to execute. This item code is meaningful only for output execution queues.
OWNER_UIC	String	The owner UIC of the specified queue.
PAGE_SETUP_MODULES	String	The names of the text modules to be extracted from the device control library and copied to the printer before each page of the specified form is printed. This item code is meaningful only for output execution queues.
PARAMETER_1 through PARAMETER_8	String	The value of the user-defined parameters that in batch jobs become the value of the DCL symbol P1 through P8 respectively.
PENDING_JOB_BLOCK_COUNT	Integer	The total number of blocks for all pending jobs in the queue (valid only for output execution queues).
PENDING_JOB_COUNT	Integer	The number of jobs in the queue in a pending state.
PENDING_JOB_REASON	Integer	The reason that the job is in a pending state.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
PEND_CHAR_MISMATCH	String	"TRUE" or "FALSE" to indicate whether job requires characteristics that are not available on the execution queue.
PEND_JOB_SIZE_MAX	String	"TRUE" or "FALSE" to indicate whether block size of job exceeds the upper block limit of the execution queue.
PEND_JOB_SIZE_MIN	String	"TRUE" or "FALSE" to indicate whether block size of job is less than the lower limit of the execution queue.
PEND_LOWERCASE_ MISMATCH	String	"TRUE" or "FALSE" to indicate whether job requires lowercase printer.
PEND_NO_ACCESS	String	"TRUE" or "FALSE" to indicate whether owner of job does not have access to the execution queue.
PEND_QUEUE_BUSY	String	"TRUE" or "FALSE" to indicate whether job is pending because the number of jobs currently executing on the queue equals the job limit for the queue.
PEND_QUEUE_STATE	String	"TRUE" or "FALSE" to indicate whether job is pending because the execution queue is not in a running, open state.
PEND_STOCK_MISMATCH	String	"TRUE" or "FALSE" to indicate whether the stock type required by the job's form does not match the stock type of the form mounted on the execution queue.
PRIORITY	Integer	The scheduling priority of the specified job.
PROCESSOR	String	The name of the symbiont image that executes print jobs initiated from the specified queue.
PROTECTION	String	The specified queue's protection mask.
QUEUE_ACL_SPECIFIED	String	"TRUE" or "FALSE" to indicate whether an access control list has been specified for the queue.
QUEUE_ALIGNING	String	"TRUE" or "FALSE" to indicate whether queue prints a specified amount of output so that paper can be properly aligned.
QUEUE_BATCH	String	"TRUE" or "FALSE" to indicate whether queue is a batch queue or a generic batch queue.
QUEUE_CLOSED	String	"TRUE" or "FALSE" to indicate whether queue is closed and will not accept new jobs until the queue is put in an open state.
QUEUE_CPU_DEFAULT	String	"TRUE" or "FALSE" to indicate whether a default CPU time limit has been specified for all jobs in the queue.
QUEUE_CPU_LIMIT	String	"TRUE" or "FALSE" to indicate whether a maximum CPU time limit has been specified for all jobs in the queue.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_FILE_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede each file in each job initiated from the queue.
QUEUE_FILE_BURST_ONE	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether flag page precedes each file in each job initiated from the queue.
QUEUE_FILE_FLAG_ONE	String	"TRUE" or "FALSE" to indicate whether flag page precedes only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether output symbiont paginates output for each job initiated from this queue. The output symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.
QUEUE_FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page follows each file in each job initiated from the queue.
QUEUE_FILE_TRAILER_ONE	String	"TRUE" or "FALSE" to indicate whether trailer page follows only the last copy of the last file in each job initiated from the queue.
QUEUE_FLAGS	Integer	The processing options that have been selected for the specified queue.
QUEUE_GENERIC	String	"TRUE" or "FALSE" to indicate whether the queue is a generic queue.
QUEUE_GENERIC_SELECTION	String	"TRUE" or "FALSE" to indicate whether the queue is an execution queue that can accept work from a generic queue.
QUEUE_IDLE	String	"TRUE" or "FALSE" to indicate whether queue prints a specified amount of output so that paper can be properly aligned.
QUEUE_JOB_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede each job initiated from the queue.
QUEUE_JOB_FLAG	String	"TRUE" or "FALSE" to indicate whether a flag page precedes each job initiated from the queue.
QUEUE_JOB_SIZE_SCHED	String	"TRUE" or "FALSE" to indicate whether jobs initiated from the queue are scheduled according to size, with the smallest job of a given priority processed first. (Meaningful only for output queues.)
QUEUE_JOB_TRAILER	String	"TRUE" or "FALSE" to indicate whether a trailer page follows each job initiated from the queue.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_LOWERCASE	String	"TRUE" or "FALSE" to indicate whether queue is associated with a printer that can print both uppercase and lowercase characters.
QUEUE_NAME	String	The name of the specified queue or the name of the queue that contains the specified job.
QUEUE_PAUSED	String	"TRUE" or "FALSE" to indicate whether execution of all current jobs in the queue is temporarily halted.
QUEUE_PAUSING	String	"TRUE" or "FALSE" to indicate whether queue is temporarily halting execution. Currently executing jobs are completing; temporarily, no new jobs can begin executing.
QUEUE_PRINTER	String	"TRUE" or "FALSE" to indicate whether the queue is a printer queue.
QUEUE_RECORD_BLOCKING	String	"TRUE" or "FALSE" to indicate whether the symbiont is permitted to concatenate, or block together, the output records it sends to the output device.
QUEUE_REMOTE	String	"TRUE" or "FALSE" to indicate whether queue is assigned to a physical device that is not connected to the local node.
QUEUE_RESETTING	String	"TRUE" or "FALSE" to indicate whether queue is resetting and stopping.
QUEUE_RESUMING	String	"TRUE" or "FALSE" to indicate whether queue is restarting after pausing.
QUEUE_RETAIN_ALL	String	"TRUE" or "FALSE" to indicate whether all jobs initiated from the queue remain in the queue after they finish executing. Completed jobs are marked with a completion status.
QUEUE_RETAIN_ERROR	String	"TRUE" or "FALSE" to indicate whether only jobs that do not complete successfully are retained in the queue.
QUEUE_SERVER	String	"TRUE" or "FALSE" to indicate whether queue processing is directed to a server symbiont.
QUEUE_STALLED	String	"TRUE" or "FALSE" to indicate whether physical device to which queue is assigned is stalled; that is, the device has not completed the last I/O request submitted to it.
QUEUE_STARTING	String	"TRUE" or "FALSE" to indicate whether queue is starting.
QUEUE_STATUS	Integer	The specified queue's status flags.
QUEUE_STOPPED	String	"TRUE" or "FALSE" to indicate whether queue is stopped.
QUEUE_STOPPING	String	"TRUE" or "FALSE" to indicate whether queue is stopping.

Table DCL-8 (Cont.): F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_SWAP	String	"TRUE" or "FALSE" to indicate whether jobs initiated from the queue can be swapped.
QUEUE_TERMINAL	String	"TRUE" or "FALSE" to indicate whether the queue is a generic queue that can place jobs only in terminal queues.
QUEUE_UNAVAILABLE	String	"TRUE" or "FALSE" to indicate whether physical device to which queue is assigned is not available.
QUEUE_WSDEFAULT	String	"TRUE" or "FALSE" to indicate whether default working set size is specified for each job initiated from the queue.
QUEUE_WSEXTENT	String	"TRUE" or "FALSE" to indicate whether working set extent is specified for each job initiated from the queue.
QUEUE_WSQUOTA	String	"TRUE" or "FALSE" to indicate whether working set quota is specified for each job initiated from the queue.
REQUEUE_QUEUE_NAME	String	The name of the queue to which the specified job is reassigned.
RESTART_QUEUE_NAME	String	The name of the queue in which the job will be placed if the job is restarted.
RETAINED_JOB_COUNT	Integer	The number of jobs in the queue retained after successful completion plus those retained on error.
SCSNODE_NAME	String	The 6-byte name of the VAX node on which jobs initiated from the specified queue execute. The node name matches the value of the SYSGEN parameter SCSNODE for the target node.
SUBMISSION_TIME	String	The time at which the specified job was submitted to the queue.
TIMED_RELEASE_JOB_COUNT	Integer	The number of jobs in the queue on hold until a specified time.
UIC	String	The UIC of the owner of the specified job.
USERNAME	String	The user name of the owner of the specified job.
WSDEFAULT	Integer	The default working set size specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.
WSEXTENT	Integer	The working set extent specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.
WSQUOTA	Integer	The working set quota for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.

example

```
$ BLOCKS = F$GETQUI("DISPLAY_ENTRY", "JOB_SIZE", 1347)
```

In this example, the F$GETQUI lexical function is used to obtain the size in blocks of print job 1347. The value returned reflects the total number of blocks occupied by the files associated with the job.

F$GETSYI

Invokes the $GETSYI system service to return status and identification information about the local system (or about a node in the local VAXcluster, if your system is part of a VAXcluster).

format

F$GETSYI*(item [,node])*

arguments

item

Indicates the type of information to be reported about the local node (or about another node in your VAXcluster, if your system is part of a VAXcluster). Specify the item as a character string expression. You can specify the items in Table DCL-9 only for your local node; you cannot specify the node argument with these items. You can specify these items whether or not you are in a VAXcluster.

You can specify the items in Table DCL-10 for either your local node or for another node in your VAXcluster. The information in this table is returned for your local node if you do not specify the node argument; the information is returned for the specified node if you include the node argument. Your system must be a member of a VAXcluster in order to specify the items in this table, except for CLUSTER_MEMBER.

node

Specifies the node in your VAXcluster for which information is to be returned. Specify the node as a character string expression. (This argument can be specified only if your system is part of a VAXcluster.)

You can request information about another node in your VAXcluster only when you specify an item from Table DCL-10. If you do not specify a node, the default is the current node. You cannot use wildcards to specify the node argument with the F$GETSYI function (as you can with the $GETSYI system service).

description

The F$GETSYI returns information on the items that can be specified with the $GETSYI system service.

Table DCL-9 lists the items you can specify with the F$GETSYI lexical function to get information about your local node. Table DCL-10 lists the items you can specify to get information about either your local node, or another node in your VAXcluster.

Table DCL-9: F$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
ARCHFLAG	String	Architecture flags for the system.
BOOTTIME	String	The time the system was booted.
CHARACTER_EMULATED	String	"TRUE" or "FALSE" to indicate whether the character string instructions are emulated on the CPU.
CPU	Integer	The processor type, as represented in the processor's SID register. For example, the integer 1 represents a VAX-11/780 and the integer 6 represents a VAX 8530, VAX 8550, VAX 8700 and VAX 8800.
DECIMAL_EMULATED	String	"TRUE" or "FALSE" to indicate whether the decimal string instructions are emulated on the CPU.
D_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the D_floating instructions are emulated on the CPU.
ERRORLOGBUFFERS	Integer	Number of system pages in use as buffers for error logging.
F_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the F_floating instructions are emulated on the CPU.
G_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the G_floating instructions are emulated on the CPU.
PAGEFILE_FREE	Integer	Number of free pages in the currently installed paging files.
PAGEFILE_PAGE	Integer	Number of pages in the currently installed paging files.
SID	Integer	System identification register.

Table DCL-9 (Cont.): F$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
SWAPFILE_FREE	Integer	Number of free pages in the currently installed swapping files.
SWAPFILE_PAGE	Integer	Number of pages in the currently installed swapping files.
VERSION	String	Version of VMS in use (8-character string filled with trailing blanks).

Table DCL-10: F$GETSYI Items for the Local Node or for Other Nodes in the VAXCluster

Item	Return Type	Information Returned
ACTIVECPU_CNT	Integer	The count of CPUs actively participating in the current boot of the SMP system.
AVAILCPU_CNT	Integer	The count of CPUs recognized in the system.
CLUSTER_FSYSID	String	System identification number for first node to boot in the VAXcluster (the founding node). This number is returned as a character string containing a hexadecimal number.
CLUSTER_FTIME	String	The time when the first node in the VAXcluster was booted.
CLUSTER_MEMBER	String	"TRUE" or "FALSE" if the node is a member of the local VAXcluster.
CLUSTER_NODES	Integer	Total number of nodes in the VAXcluster, as an integer.
CLUSTER_QUORUM	Integer	Total quorum for the VAXcluster.
CLUSTER_VOTES	Integer	Total number of votes in the VAXcluster.
CONTIG_GBLPAGES	Integer	Total number of free, contiguous global pages.
FREE_GBLPAGES	Integer	Current count of free global pages.
FREE_GBLSECTS	Integer	Current count of free global section table entries.
HW_MODEL	Integer	An integer that identifies the node's VAX model type.
HW_NAME	String	The VAX model name.
NODENAME	String	Node name.
NODE_AREA	Integer	The VAX DECnet area for the target node.
NODE_CSID	String	The CSID of the specified node, as a string containing a hexadecimal number. The CSID is a form of system identification.

Table DCL-10 (Cont.): F$GETSYI Items for the Local Node or for Other Nodes in the VAXCluster

Item	Return Type	Information Returned
NODE_HWTYPE	String	Hardware type of the specified node.
NODE_HWVERS	String	Hardware version of the specified node.
NODE_NUMBER	Integer	The VAX DECnet number for the specified node.
NODE_QUORUM	Integer	Quorum that the node has.
NODE_SWINCARN	String	Software incarnation number for the specified node. This number is returned as a string containing a hexadecimal number.
NODE_SWTYPE	String	Type of operating system software used by the specified node.
NODE_SWVERS	String	Software version of the specified node.
NODE_SYSTEMID	String	System identification number for the specified node. This number is returned as a string containing a hexadecimal number.
NODE_VOTES	Integer	Number of votes that the node has.
SCS_EXISTS	String	"TRUE" or "FALSE" to indicate whether the system communication subsystem (SCS) is currently loaded on a VAX node.

example

```
$ MEM = F$GETSYI("CLUSTER_MEMBER", "LONDON")
$ SHOW SYMBOL MEM
  MEM = "TRUE"
```

This example uses the F$GETSYI function to determine whether the node LONDON is a member of the local VAXcluster. The "TRUE" indicates that the remote node LONDON is a member of the VAXcluster.

F$IDENTIFIER

Converts an alphanumeric identifier to its integer equivalent, or converts an integer identifier to its alphanumeric equivalent. An identifier is a name or number that identifies a category of users of a data resource. The system uses identifiers to determine a user's access to a resource.

format

F$IDENTIFIER*(identifier,conversion-type)*

arguments

identifier

Specifies the identifier to be converted. Specify the identifier as an integer expression if you are converting an integer to a name. Specify the identifier as a character string expression if you are converting a name to an integer. The F$IDENTIFIER function does not convert letters in the identifier to uppercase. Therefore, you must specify the identifier the same way it is defined in the rights database.

conversion-type

Indicates the type of conversion to be performed. If the identifier argument is alphanumeric, specify the translation argument as a character string containing "NAME_TO_NUMBER". If the identifier argument is numeric, specify the translation argument as a character string containing "NUMBER_TO_NAME".

example

```
$ UIC_INT= F$IDENTIFIER("SLOANE","NAME_TO_NUMBER")
$ SHOW SYMBOL UIC_INT
  UIC_INT = 15728665   Hex = 00F00019  Octal = 00074000031
$ UIC = F$FAO("!%U",UIC_INT)
$ SHOW SYMBOL UIC
  UIC = [360,031]
```

This example uses the F$IDENTIFIER to convert the member identifier from the UIC [MANAGERS,SLOANE] to an integer. The F$IDENTIFIER function shows that the member identifier SLOANE is equivalent to the integer 15728665. Note that you must specify the identifier SLOANE using uppercase letters.

To convert this octal number to a standard numeric UIC, use the F$FAO function with the !%U directive. (This directive converts a longword to a UIC in named format.) In this example, the member identifier SLOANE is equivalent to the numeric UIC [360,031].

F$INTEGER

Returns the integer equivalent of the result of the specified expression.

format

F$INTEGER*(expression)*

argument

expression

Specifies the expression to be evaluated. Specify either an integer or a character string expression. If you specify an integer expression, the F$INTEGER function evaluates the expression and returns the result. If you specify a string expression, the F$INTEGER function evaluates the expression, converts the resulting string to an integer, and returns the result. If the string contains characters that do not form a valid integer, the F$INTEGER function returns the integer 1 if the string begins with T, t, Y, or y. The function returns the integer 0 if the string begins with any other character.

example

```
$ A = "23"
$ B = F$INTEGER("-9" + A)
$ SHOW SYMBOL B
  B = -923 Hex=FFFFFC65 Octal=176145
```

This example shows how to use the F$INTEGER function to equate a symbol to the integer value returned by the function.

The F$INTEGER function in the above example returns the integer equivalent of the string expression ("–9" + A). First, the F$INTEGER function evaluates the string expression by concatenating the string literal "–9" with the string literal "23". Note that the value of the symbol A is automatically substituted in a string expression. Also note that the plus sign (+) is a string concatenation operator since both arguments are string literals.

After the string expression is evaluated, the F$INTEGER function converts the resulting character string ("–923") to an integer, and returns the value –923. This integer value is assigned to the symbol B.

F$LENGTH

Returns the length of the specified character string.

format

F$LENGTH*(string)*

argument

string

Specifies the character string whose length is being determined. Specify the string argument as a character string expression.

example

```
$ MESSAGE = F$MESSAGE(%X1C)
$ SHOW SYMBOL MESSAGE
  MESSAGE = "%SYSTEM-F-EXQUOTA, exceeded quota"
$ STRING_LENGTH = F$LENGTH(MESSAGE)
$ SHOW SYMBOL STRING_LENGTH
  STRING_LENGTH = 33   Hex = 00000021  Octal = 000041
```

The first assignment statement uses the F$MESSAGE function to return the message that corresponds to the hexadecimal value 1C. The message is returned as a character string and is assigned to the symbol MESSAGE.

The F$LENGTH function is then used to return the length of the character string assigned to the symbol MESSAGE. You do not need to use quotation marks when you use the symbol MESSAGE as an argument for the F$LENGTH function. (Quotation marks are not used around symbols in character string expressions.)

The F$LENGTH function returns the length of the character string and assigns it to the symbol STRING_LENGTH. At the end of the example, the symbol STRING_LENGTH has a value equal to the number of characters in the value of the symbol named MESSAGE, that is, 33.

F$LOCATE

Locates a specified portion of a character string and returns as an integer the offset of the first character. (An offset is the position of a character or a substring relative to the begining of the string. The first character in a string is always offset position 0 from the beginning of the string.) If the substring is not found, F$LOCATE returns the length (the offset of the last character in the character string plus one) of the searched string.

format

F$LOCATE*(substring,string)*

arguments

substring

The character string, specified in the string argument, that you want to locate within the string.

string

The character string to be edited by F$LOCATE.

example

```
$ INQUIRE TIME "Enter time"
$ IF F$LOCATE(":",TIME) .EQ. F$LENGTH(TIME) THEN -
  GOTO NO_COLON
```

This section of a command procedure compares the results of the F$LOCATE and F$LENGTH functions to see if they are equal. This technique is commonly used to determine whether a character or substring is contained in a string.

In the example, the INQUIRE command prompts for a time value and assigns the user-supplied time to the symbol TIME. The IF command checks for the presence of a colon in the string entered in response to the prompt. If the value returned by the F$LOCATE function equals the value returned by the F$LENGTH function, the colon is not present. You use the .EQ. operator (rather than .EQS.) because the F$LOCATE and F$LENGTH functions return integer values.

Note that quotation marks are used around the substring argument, the colon, because it is a string literal. However, the symbol TIME does not require quotation marks because it is automatically evaluated as a string expression.

F$LOGICAL

Translates a logical name and returns the equivalence name string (or a null string if no match is found). The translation is not iterative; the equivalence string is not checked to determine whether it is a logical name.

As of VMS Version 4.0, the F$LOGICAL lexical function is superseded by F$TRNLNM. DIGITAL recommends using the F$TRNLNM lexical function. See F$TRNLNM for a complete description of the F$TRNLNM lexical function.

format

F$LOGICAL*(logical-name)*

F$MESSAGE

Returns as a character string the facility, severity, identification, and text associated with the specified system status code.

format

F$MESSAGE*(status-code)*

argument

status-code
The status code for which you are requesting error message text. You must specify the status code as an integer expression.

example

```
$ ERROR_TEXT = F$MESSAGE(%X1C)
$ SHOW SYMBOL ERROR_TEXT
  ERROR_TEXT = "%SYSTEM-F-EXQUOTA, exceeded quota"
```

This example shows how to use the F$MESSAGE function to determine the message associated with the status code %X1C. The F$MESSAGE function returns the message string, which is assigned to the symbol ERROR_TEXT.

F$MODE

Returns a character string showing the mode in which a process is executing. Returns the character string "INTERACTIVE" for interactive processes. If the process is noninteractive, the character string "BATCH", "NETWORK" or "OTHER" is returned. The return string always contains uppercase letters. The F$MODE function has no arguments, but must be followed by parentheses.

format

F$MODE()

arguments

None.

example

```
$ IF F$MODE() .NES. "INTERACTIVE" THEN GOTO NON_INT_DEF
$ INTDEF:          ! Commands for interactive terminal sessions
   .
   .
   .
$ EXIT
$ NON_INT_DEF:          !Commands for non-interactive processes
   .
   ..
   .
```

This example shows the beginning of a login.com file that has two sets of initialization commands: one for interactive mode and one for noninteractive mode (including batch and network jobs). The IF command compares the character string returned by F$MODE with the character string INTERACTIVE; if they are not equal, control branches to the label NON_INT_DEF. If the character strings are equal, the statements following the label INTDEF are executed and the procedure exits before the statements at NON_INT_DEF.

F$PARSE

Invokes the $PARSE RMS service to parse a file specification and return as a character string either the expanded file specification or the particular file specification field that you request.

format

F$PARSE*(file-spec [,default-spec] [,related-spec] [,field] [,parse-type])*

arguments

file-spec
Specifies a character string containing the file specification to be parsed. The file specification can contain wildcard characters. If you use a wildcard character, the file specification returned by the F$PARSE function contains the wildcard.

default-spec
Specifies a character string containing the default file specification. The fields in the default file specification are substituted in the output string if a particular field in the file-spec argument is missing. You can make further substitutions in the file-spec argument by using the related-spec argument.

related-spec
Specifies a character string containing the related file specification. The fields in the related file specification are substituted in the output string if a particular field is missing from both the file-spec and default-spec arguments.

field

Specifies a character string containing the name of a field in a file specification. Specifying the field argument causes F$PARSE to return a specific portion of a file specification.

Specify one of the following field names (do not abbreviate):

NODE	Node name
DEVICE	Device name
DIRECTORY	Directory name
NAME	File name
TYPE	File type
VERSION	File version number

parse-type

The type of parsing to be performed. By default, the F$PARSE function verifies that the directory in the file specification exists on the device in the file specification. Note that the device and directory can be explicitly given in one of the arguments, or can be provided by default. Also, by default the F$PARSE function translates logical names if they are provided in any of the arguments. You can change how the F$PARSE function parses a file specification by using one of the following keywords:

NO_CONCEAL	Ignores the "conceal" attribute in the translation of a logical name as part of the file specification; that is, logical name translation does not end when a concealed logical name is encountered.
SYNTAX_ONLY	The syntax of the file specification is checked without verifying that the specified directory exists on the specified device.

description

When you use the F$PARSE function, you can omit those optional arguments to the right of the last argument you specify. However, you must include commas as placeholders if you omit optional arguments to the left of the last argument you specify. If you omit the device and directory names in the file-spec argument, the F$PARSE function supplies defaults, first from the default-spec argument and second from the related-spec argument. If names are not provided by these arguments, the F$PARSE function uses your current default disk and directory. If you omit the file name, file type, or version number, the F$PARSE function supplies defaults, first from the default-spec argument and second from the related-spec argument. If names are not provided by these arguments, the F$PARSE function returns a null specification for these fields.

example

```
$ SET DEF DISK2:[FIRST]
$ SPEC = F$PARSE("JAMES.MAR","[ROOT]",,,"SYNTAX_ONLY")
$ SHOW SYMBOL SPEC
  SPEC = "DISK2:[ROOT]JAMES.MAR;"
```

In this example, the F$PARSE function returns the expanded file specification for the file JAMES.MAR. The example uses the SYNTAX_ONLY keyword to request that F$PARSE should check the syntax, but should not verify that the [ROOT] directory exists on DISK2.

The default device and directory are DISK2:[FIRST]. Because the directory name [ROOT] is specified as the default-spec argument in the assignment statement, it is used as the directory name in the output string. Note that the default device returned in the output string is DISK2, and the default version number for the file is null. You must place quotation marks around the arguments JAMES.MAR and ROOT because they are string literals.

If you had not specified syntax-only parsing, and [ROOT] were not on DISK2, a null string would have been returned.

F$PID

The F$PID function returns a process identification (PID) number as a character string and updates the context symbol to point to the current position in the system's process list. The PIDs returned by the F$PID function depend on the privilege of your process. If you have GROUP privilege, the F$PID function returns PIDs of processes in your group. If you have WORLD privilege, the F$PID function returns PIDs of all processes on the system. If you lack GROUP or WORLD privileges, the F$PID function returns only your process PID.

format

F$PID*(context-symbol)*

argument

context-symbol

Specifies a symbol that DCL uses to store a pointer into the system's list of processes. The F$PID function uses this pointer to return a PID. Specify the context-symbol by using a symbol. The first time you use the F$PID function in a command procedure, you should use a symbol that is either undefined or equated to the null string ("").

example

```
$ CONTEXT = ""
$ START:
$     PID = F$PID(CONTEXT)
$     IF PID .EQS. "" THEN EXIT
$     SHOW SYMBOL PID
$     GOTO START
```

This command procedure uses the F$PID function to display a list of PIDs. The assignment statement declares the symbol CONTEXT, which is used as the context-symbol argument for the F$PID function. Because CONTEXT is equated to a null string, the F$PID function returns the first PID in the process list that it has the privilege to access.

The PIDs displayed by this command procedure depend on the privilege of your process. When run with GROUP privilege, the PIDs of users in your group are displayed. When run with WORLD privilege, the PIDS of all users on the system are displayed. Without GROUP or WORLD privilege, only your PID is displayed.

F$PRIVILEGE

Returns a value of either "TRUE" or "FALSE", depending on whether your current process privileges match those specified in the argument. You can specify either the positive or negative version of a privilege.

format

F$PRIVILEGE*(priv-states)*

argument

priv-states
A character string containing a privilege or a list of privileges separated by commas. For a list of process privileges, see Table 4-1 in the *VMS System Manager's Manual*. Specify any one of the process privileges except [NO]ALL.

description

Use the F$PRIVILEGE function to identify your current process privileges.

If "NO" precedes the privilege, the privilege must be disabled in order for the function to return a value of "TRUE". The F$PRIVILEGE function checks each of the keywords in the specified list, and if the result for any one is false, the string "FALSE" is returned.

example

```
$ PROCPRIV = F$PRIVILEGE("OPER,GROUP,TMPMBX,NONETMBX")
$ SHOW SYMBOL PROCPRIV
  PROCPRIV = "FALSE"
```

The F$PRIVILEGE function is used to test whether the process has OPER, USER, TMPMBX, and NETMBX privileges.

The process in this example has OPER, GROUP, TMPMBX, and NETMBX privileges. Therefore, a value of "FALSE" is returned because the process has NETMBX privilege, but NONETMBX was specified in the priv-states list. Although the Boolean result for the other three keywords is true, the entire expression is declared false because the result for NONETMBX was false.

F$PROCESS

Obtains the current process name string. The F$PROCESS function has no arguments, but must be followed by parentheses.

format

F$PROCESS()

arguments

None.

example

```
$ NAME = F$PROCESS()
$ SHOW SYMBOL NAME
  NAME = "MARTIN"
```

In this example, the F$PROCESS function returns the current process name and assigns it to the symbol NAME.

F$SEARCH

Invokes the $SEARCH RMS service to search a directory file and return the full file specification for a file you specify.

format

F$SEARCH*(file-spec[,stream-id])*

arguments

file-spec

Specifies a character string containing the file specification to be searched for. If the device or directory names are omitted, the defaults from your current default disk and directory are used. The F$SEARCH function does not supply defaults for a file name or type. If the version is omitted, the specification for the file with the highest version number is returned. If the file-spec argument contains wildcards, each time F$SEARCH is called, the next file specification that agrees with the file-spec argument is returned. A null string is returned after the last file specification that agrees with the file-spec argument.

stream-id

A positive integer representing the search stream identification number. The search stream identification number is used to maintain separate search contexts when you use the F$SEARCH function more than once and when you supply different file-spec arguments. If you omit *stream-id*, the F$SEARCH function assumes an implicit single search stream. That is, the F$SEARCH function starts searching at the beginning of the directory file each time you specify a different file-spec argument.

example

```
$ START:
$    COM = F$SEARCH ("*.COM;*",1)
$    DAT = F$SEARCH ("*.DAT;*",2)
$    SHOW SYMBOL COM
$    SHOW SYMBOL DAT
$    IF (COM.EQS. "") .AND. (DAT.EQS. "") THEN EXIT
$    GOTO START
```

This command procedure searches the default disk and directory for both COM and DAT files. Note that the stream-id is specified for each F$SEARCH function so that the context for each search is maintained.

The first F$SEARCH function starts searching from the top of the directory file for a file with a type of COM. When it finds a COM file, a pointer is set to maintain the search context. When the F$SEARCH function is used the second time, it again starts searching from the top of the directory file for a file with a type of DAT. When the procedure loops back to the label START, the stream-id argument allows each F$SEARCH function to start searching in the correct place in the directory file. After all versions of COM and DAT files are returned, the procedure exits.

F$SETPRV

Invokes the $SETPRV system service to enable or disable specified user privileges. The F$SETPRV function returns a list of keywords indicating user privileges; this list shows the status of the specified privileges before F$SETPRV was executed. Your process must be authorized to set the specified privilege.

format

F$SETPRV*(priv-states)*

argument

priv-states
A character string defining a privilege or a list of privileges separated by commas. For a list of process privileges, see Table 4-1 in the *VMS System Manager's Manual*.

example

```
$ OLDPRIV = F$SETPRV("OPER,NOTMPMBX")
$ SHOW SYMBOL OLDPRIV
  OLDPRIV = "NOOPER,TMPMBX"
```

In this example, the process is authorized to change the OPER and TMPMBX privileges. The F$SETPRV function enables the OPER privilege and disables the TMPMBX privilege. In addition, the F$SETPRV function returns the keywords NOOPER and TMPMBX, showing the state of these privileges before they were changed.

You must place quotation marks around the list of privilege keywords because it is a string literal.

F$STRING

Returns the string that is equivalent to the specified expression.

format

F$STRING*(expression)*

argument

expression
The integer or string expression to be evaluated. If you specify an integer expression, the F$STRING expression evaluates the expression, converts the resulting integer to a string, and returns the result. If you specify a string expression, the F$STRING expression evaluates the expression and returns the result. When converting an integer to a string, the F$STRING function uses decimal representation and omits leading zeroes. When converting

a negative integer, the F$STRING function places a minus sign at the beginning string representation of the integer.

example

```
$ A = 5
$ B = F$STRING(-2 + A)
$ SHOW SYMBOL B
  B = "3"
```

The F$STRING function in this example converts the result of the integer expression (–2 + A) to the numeric string, "3". First, the F$STRING function evaluates the expression (–2 + A). Note that 5, the value of symbol A, is automatically substituted when the integer expression is evaluated.

After the integer expression is evaluated, the F$STRING function converts the resulting integer, 3, to the string "3". This string is assigned to the symbol B.

F$TIME

Returns as a character string the current date and time in absolute time format. The returned string has the following fixed, 23-character format:

dd-mmm-yyyy hh:mm:ss.cc

The F$TIME function has no arguments, but must be followed by parentheses.

format

F$TIME()

arguments

None.

example

```
$ OPEN/WRITE OUTFILE DATA.DAT
$ TIME_STAMP = F$TIME()
$ WRITE OUTFILE TIME_STAMP
```

This example shows how to use the F$TIME function to time-stamp a file that you create from a command procedure. OUTFILE is the logical name for the file DATA.DAT, which is opened for writing. The F$TIME function returns the current date and time string, and assigns this string to the symbol TIME_STAMP. The WRITE command writes the date and time string to OUTFILE.

F$TRNLNM

Translates a logical name and returns the equivalence name string, or the requested attributes of the logical name specified. The return value can be a character string or an integer, depending on the arguments you specify with the F$TRNLNM function. The translation is not iterative; the equivalence string is not checked to determine whether it is a logical name.

format

F$TRNLNM*(logical-name [,table] [,index] [,mode] [,case] [,item])*

arguments

logical-name
Specifies a character string containing the logical name to be translated.

table
Specifies a character string containing the logical name table or tables that the F$TRNLNM function should search to translate the logical name. The table argument must be a logical name that translates to a logical name table or to a list of table names. If you do not specify a table, the default value is LNM$DCL_LOGICAL. Unless LNM$DCL_LOGICAL has been redefined for your process, the F$TRNLNM function searches the process, job, group, and system logical name tables, in that order, and returns the equivalence name for the first match found.

index
Specifies the number of the equivalence name to be returned if the logical name has more than one translation. If you do not specify the index argument, the default is 0.

mode
Specifies a character string containing one of the following access modes for the translation: USER (default), SUPERVISOR, EXECUTIVE, or KERNEL. The F$TRNLNM function starts by searching for a logical name created with the access mode specified in the mode argument. If it does not find a match, the F$TRNLNM function searches for the name created with each inner access mode and returns the first match found.

case
Specifies the type of case translation to be performed. Specify the case argument as either of the following character strings: CASE_BLIND (default) or CASE_SENSITIVE. If the translation is case blind, the F$TRNLNM function first searches for a logical name with characters of the same case as the name argument. If no match is found, the F$TRNLNM function searches for an uppercase version of the name argument and the logical names it is searching. The result of the first successful translation is returned. If the translation is case sensitive, the F$TRNLNM function searches only for a

logical name with characters of the same case as the name argument. The F$TRNLNM function returns a null string if no exact match is found.

item

A character string containing the type of information that F$TRNLNM should return about the specified logical name. Specify one of the following items:

Item	Return Type	Information Returned
ACCESS_MODE	String	One of the following access modes associated with the logical name: "USER", "SUPERVISOR", "EXECUTIVE", "KERNEL".
CONCEALED	String	Either "TRUE" or "FALSE" to indicate whether the CONCEALED attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created.
CONFINE	String	Either "TRUE" or "FALSE" to indicate whether the logical name is confined. If the logical name is confined (TRUE), then the name is not copied to subprocesses. If the logical name is not confined (FALSE), then the name is copied to subprocesses.
CRELOG	String	"TRUE" or "FALSE" to indicate whether the logical name was created with the $CRELOG system service or with the $CRELNM system service, using the CRELOG attribute.
LENGTH	Integer	Length of the equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F$TRNLNM function returns the length of the name specified by the index argument.
MAX_INDEX	Integer	The largest index defined for the logical name. The index shows how many equivalence names are associated with a logical name. The index is zero based; that is, the index 0 refers to the first name in a list of equivalence names.
NO_ALIAS	String	Either "TRUE" or "FALSE" to indicate whether the logical name has the NO_ALIAS attribute. The NO_ALIAS attribute means that a logical name must be unique within outer access mode.
TABLE	String	Either "TRUE" or "FALSE" to indicate whether the logical name is the name of a logical name table.
TABLE_NAME	String	Name of the table where the logical name was found.

Item	Return Type	Information Returned
TERMINAL	String	Either "TRUE" or "FALSE" to indicate whether the TERMINAL attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created. The TERMINAL attribute indicates that the logical name is not a candidate for iterative translation.
VALUE	String	Default. The equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F$TRNLNM function returns the name specified by the index argument.

example

```
$ DEFINE/TABLE=LNM$GROUP TERMINAL 'F$TRNLNM("SYS$OUTPUT")'
```

This example shows a line from a command procedure that (1) uses the F$TRNLNM function to determine the name of the current output device and (2) creates a group logical name table entry based on the equivalence string.

You must enclose the argument SYS$OUTPUT in quotation marks because it is a character string.

Also, in this example you must enclose the F$TRNLNM function in single quotes to force the lexical function to be evaluated. Otherwise, the DEFINE command does not automatically evaluate the lexical function.

F$TYPE

Returns the data type of a symbol. The string "INTEGER" if the symbol is equated to an integer, or if the symbol is equated to a string whose characters form a valid integer. If the symbol is equated to a character string whose characters do not form a valid integer, the F$TYPE function returns the string "STRING". If the symbol is undefined, a null string is returned.

format

F$TYPE*(symbol-name)*

argument

symbol
Specifies a character string or integer that references the name of the symbol to be evaluated.

example

```
$ NUM = "52"
$ TYPE = F$TYPE(NUM)
$ SHOW SYMBOL TYPE
  TYPE = "INTEGER"
```

This example uses the F$TYPE function to determine the data type of the symbol NUM. NUM is equated to the character string "52". Because the characters in the string form a valid integer, the F$TYPE function returns the string "INTEGER".

F$USER

Returns the current user identification code (UIC) in named format as a character string. The F$USER function has no arguments, but must be followed by parentheses.

format

F$USER()

arguments

None.

example

```
$ UIC = F$USER()
$ SHOW SYMBOL UIC
  UIC = "[GROUP6,JENNIFER]"
```

In this example the F$USER function returns the current user identification code and assigns it to the symbol UIC.

F$VERIFY

Returns an integer value indicating whether the procedure verification setting is currently on or off. If used with arguments, the F$VERIFY function can turn the procedure and image verification settings on or off. You must include the parentheses after the F$VERIFY function whether or not you specify arguments.

format

F$VERIFY*([procedure-value] [,image-value])*

arguments

procedure-value

An integer expression with a value of 1 to turn procedure verification on, or 0 to turn procedure verification off. When procedure verification is on, each DCL command line in the command procedure is displayed on the output device. Procedure verification allows you to verify that each command is executing correctly.

image-value

An integer expression with a value of 1 to turn image verification on, or 0 to turn image verification off. When image verification is on, data lines in the command procedure are displayed on the output device.

description

Using the F$VERIFY function in command procedures allows you to test the current procedure verification setting. For example, a command procedure can save the current procedure verification setting before changing it and then later restore the setting.

If you do not specify any arguments, neither of the verification settings is changed. If you specify only the procedure-value argument, both procedure and image verification are turned on (if the value is 1) or off (if the value is 0). If you specify both arguments, procedure and image verification are turned on or off independently. If you specify the image-value argument alone, only image verification is turned on or off. If you specify the image-value argument alone, you must precede the argument with a comma.

example

```
$ SAVE_PROC_VERIFY = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ SAVE_IMAGE_VERIFY = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
    .
    .
    .
$ TEMP = F$VERIFY(SAVE_PROC_VERIFY, SAVE_IMAGE_VERIFY)
```

This example shows an excerpt from a command procedure. The first assignment statement assigns the current procedure verification setting to the symbol SAVE_PROC_VERIFY. The second assignment statement assigns the current image verification setting to the symbol SAVE_IMAGE_VERIFY.

Then, the SET NOVERIFY command disables procedure and image verification. Later, the F$VERIFY function resets the verification settings, using the original values (equated to the symbols SAVE_PROC_VERIFY and SAVE_IMAGE_VERIFY). The symbol TEMP contains the procedure verification before it is changed with the F$VERIFY function. (In this example the value of TEMP is not used.)

LIBRARY

Invokes the Librarian Utility to create, modify, or describe an object, macro, help, text, or shareable image library.

format

LIBRARY *library-file-spec [input-file-spec[,...]]*

LINK

Invokes the VMS Linker to link one or more object modules into a program image and defines execution characteristics of the image.

format

LINK *file-spec[,...]*

parameter

file-spec[,...]
Specifies one or more input files (wildcard characters not allowed). Input files may be object modules, libraries to be searched for external references or from which specific modules are to be included, shareable images to be included in the output image or option files to be read by the linker. Separate multiple input file specifications with commas (,) or plus signs (+). In either case, the linker creates a single image file.

If you omit the file type in an input file specification, the linker supplies default file types, based on the nature of the file. For object modules, file type OBJ is assumed.

qualifiers

/BRIEF
Requests the linker to produce a brief map file; valid only with the /MAP qualifier.

/CONTIGUOUS
/NOCONTIGUOUS (default)
Controls whether the output image file is contiguous.

/CROSS_REFERENCE
/NOCROSS_REFERENCE (default)
Controls whether the map contains a symbol cross-reference list with entries for each global symbol referenced in the image, its value, and all modules in the image that refer to it.

/DEBUG[=file-spec]
/NODEBUG
If the object module contains local symbol table or traceback information, you can specify /DEBUG to include the information in the image as well. If the object module does not contain local symbol table or traceback information, only global symbols are available for symbolic debugging.

By default, the VMS Debugger is linked with the image. However, you may use the file-spec option to specify an alternate debugger (wildcard characters not allowed).

/EXECUTABLE[=file-spec]
/NOEXECUTABLE
Permits you to specify whether or not the linker creates an executable image. By default the linker creates an executable image with the same file name as the first input file and a file type of EXE but this qualifier gives you the option of assigning the image a file specification (wildcard characters not allowed). The placement of the command qualifier determines the output file specification defaults.

/FULL
Requests a full map listing; valid only with /MAP qualifier.

/HEADER
Provides a system image header when used with the /SYSTEM qualifier.

/INCLUDE=(module-name[,...])
Positional qualifier. Selects modules from the associated object module library or image library as input to the linking operation. No wildcard characters are allowed in the module name specifications. If you specify several modules, separate them with commas and enclose the list in parentheses.

/LIBRARY
Positional qualifier. Indicates that the associated input file is a library (default file type OLB) whose modules should be searched to resolve undefined symbols. You are not permitted to specify a library as the first input file unless you also specify the /INCLUDE qualifier to indicate which modules in the library are to be included in the input.

/MAP[=file-spec]
/NOMAP
Permits you to specify whether or not a memory allocation listing (map) is produced and gives you the option of assigning it a file specification. In interactive mode, the default is /NOMAP; in batch mode, the default is /MAP. You can specify the map's contents using either the /BRIEF, /FULL, or /CROSS_REFERENCE qualifiers.

/OPTIONS

Positional qualifier. Indicates that the associated input file (default file type OPT) contains a list of linking options.

/P0IMAGE

Directs the linker to create an image in P0 address space together with the stack and the VMS RMS buffers that usually go in P1 address space.

/PROTECT

When used with the /SHAREABLE qualifier, directs the linker to create a protected shareable image that can execute privileged change mode instructions even when it is linked into a nonprivileged executable image.

/SELECTIVE_SEARCH

Positional qualifier. Use this qualifier when you want the linker to omit from the output image symbol table, all symbols from the associated input object module that are not needed to resolve outstanding references.

/SHAREABLE[=file-spec]
/NOSHAREABLE

Command qualifier. By default, the linker creates an executable image. If you specify the /SHAREABLE qualifier, the linker creates a shareable image file instead. Optionally, you may designate a name for the output file; however, wildcard characters are not permitted. To specify an input shareable image, the /SHAREABLE qualifier must be used as an input file qualifier in an options file.

/SHAREABLE
/SHAREABLE=NOCOPY

Positional qualifier. Use this positional qualifier in the context of an options file only to identify an input file as a shareable image file. The keyword NOCOPY tells the linker not to bind a private copy of the shareable image to the executable image.

/SYMBOL_TABLE[=file-spec]
/NOSYMBOL_TABLE

The default is /NOSYMBOL_TABLE (do not create a symbol table). Use the /SYMBOL_TABLE qualifier when you want the linker to create a symbol table object module file (default file type STB) that contains symbol definitions for all global symbols in the image being linked. The symbol table file can be subsequently specified in LINK commands to provide the symbol definitions to other images.

When you specify /SYMBOL_TABLE, you can control the defaults applied to the output file specification. Optionally, you may designate a name for the symbol table file, but you may not use wildcard characters.

/SYSLIB
/NOSYSLIB

The default is /SYSLIB (search the system libraries). Use the /NOSYSLIB qualifier to prevent the linking operation from automatically searching the default system libraries, SYS$LIBRARY:IMAGELIB.OLB and then SYS$LIBRARY:STARLET.OLB, for unresolved references in the input files.

/SYSSHR
/NOSYSSHR

The default is /SYSSHR (search the default system shareable image library). Use the /NOSYSSHR qualifier to prevent the linking operation from automatically searching the default system shareable image library, SYS$LIBRARY:IMAGELIB.OLB, for unresolved references.

/SYSTEM[=base-address]
/NOSYSTEM

The default is /NOSYSTEM (do not produce a system image). Use the /SYSTEM qualifier to produce a system image and optionally assign it a base address. You cannot use the /SYSTEM qualifier with either the /SHAREABLE qualifier or the /DEBUG qualifier. The base address specifies where the image is to be loaded in virtual memory It can be expressed in decimal, hexadecimal, or octal format, using the radix specifiers %D, %X, or %O, respectively. The default base address is %X80000000.

/TRACEBACK (default)
/NOTRACEBACK

Default is /TRACEBACK (include traceback information in the image file to help the system trace the call stack when an error occurs). Use the /NOTRACEBACK qualifier to prevent the linker from including traceback information.

If you specify /DEBUG, /TRACEBACK is assumed.

/USERLIBRARY[=(table[,...])]
/USERLIBRARY=ALL

You use this qualifier to specify which user-defined default libraries (process, group, system or, by default, all three) the linker searches after it has searched any specified user libraries. The /NOUSERLIBRARY qualifier tells the linker not to search any user-defined default libraries.

example

```
$ LINK/MAP/FULL DRACO,CYGNUS,LYRA
```

The LINK command in this example links the modules DRACO.OBJ, CYGNUS.OBJ, and LYRA.OBJ and creates an executable image named DRACO.EXE. The /MAP and /FULL qualifiers request a full map of the image, with descriptions of each program section, lists of global symbols by name and by value, and a summary of the image characteristics. The map file is named DRACO.MAP.

LOGIN Procedure

Initiates an interactive terminal session.

format

CTRL/C

CTRL/Y

RET

qualifiers

/CLI=command-language-interpreter

Specifies the name of an alternate command language interpreter (CLI) to override the default CLI listed in the user authorization file. The CLI you specify must be located in SYS$SYSTEM and have the file type EXE.

If you do not specify a command interpreter using the /CLI qualifier and do not have a default CLI listed in the user authorization file, the system supplies a default of /CLI=DCL.

/COMMAND[=file-spec]
/NOCOMMAND

Controls whether to execute your default login command procedure when you log in. Use the /COMMAND qualifier to specify the name of an alternate login command procedure. If you specify a file name without a file type, the default file type COM is used. If you specify /COMMAND and omit the file specification, your default login command procedure is executed. By default, /COMMAND is assumed.

Use the /NOCOMMAND qualifier if you do not want your default login command procedure to be executed.

/DISK=device-name[:]

Specifies the name of a disk device to be associated with the logical device SYS$DISK for the terminal session. This specification overrides the default SYS$DISK device established in the authorization file.

/TABLES=(command-table[,...])

Specifies the name of an alternate CLI table to override the default listed in the user authorization file (UAF). This table name is considered a file specification. The default device and directory is SYS$SHARE and the default file type is EXE.

If a logical name is used, the table name specification must be defined in the system logical name table.

If the /CLI qualifier is set to DCL or MCR, the /TABLES qualifier defaults to the correct value. If the /TABLES qualifier is specified without the /CLI qualifier, the CLI specified in the user's UAF will be used.

The default is /TABLES=DCLTABLES.

example

```
RET
Username: JONES
Password:
User authorization failure
RET
Username: JONES
Password:
        Welcome to VAX/VMS Version 5.00 on node JUPITER
        Last interactive login on Tuesday, 16-AUG-1988 09:16:47.08
        Last non-interactive login on Monday, 15-AUG-1988 17:32:34.27
        1 failure since last successful login.
$
```

This example shows the "User authorization failure" message, which indicates that the password has been entered incorrectly. After successfully logging in, a message is displayed showing the number of login failures since your last successful login. This message is displayed only if one or more login failures have occurred.

LOGOUT

Terminates an interactive terminal session.

format

LOGOUT

qualifiers

/BRIEF

Prints a brief logout message (process name, date, and time) or a full logout message (a brief message plus accounting statistics).

/FULL

Requests the long form of the logout message. When you specify /FULL, the command interpreter displays a summary of accounting information for the terminal session. The default for a batch job is /FULL.

/HANGUP
/NOHANGUP

For dialup terminals, determines whether or not the phone hangs up whenever you log out. By default, the /HANGUP setting of your terminal port determines whether the line is disconnected. Your system manager determines whether you are permitted to use this qualifier.

example

```
$ LOGOUT/FULL
  HIGGINS    logged out at 15-APR-1988 14:23:45.30
Accounting information:
Buffered I/O count:          22     Peak working set size:     90
Direct I/O count:            10     Peak virtual size:         69
Page faults:                 68     Mounted volumes:            0
Charged CPU time: 0 00:01:30.50  Elapsed time:      0 04:59:02.63
```

In this example, the LOGOUT command with the /FULL qualifier displays a summary of accounting statistics for the terminal session.

MACRO

Invokes the VAX MACRO assembler to assemble one or more assembly language source files.

See the qualifier descriptions for restrictions.

format

MACRO *file-spec-list*

parameter

file-spec-list

Requests the assembly of one or more VAX MACRO assembly language source files. The file-spec-list parameter consists of one or more file specifications. For each file specification, the MACRO command supplies a default file type of MAR. You cannot include a wildcard character in a file specification.

File specifications separated by commas cause the MACRO assembler to produce an object file (and, if indicated, a listing file) for each specified file. File specifications separated by plus signs are concatenated into one input file and produce a single object file (and listing file). The MACRO assembler creates output files of one higher version than the highest version existing in the target directory.

qualifiers

/ANALYSIS_DATA[=file-spec]
/NOANALYSIS_DATA (default)

Controls whether the assembler creates an analysis data file for the VAX Source Code Analyzer (SCA), and optionally provides the file specification.

By default, the assembler does not create an analysis data file. If you specify /ANALYSIS_DATA without a file specification, the assembler creates a file with the same file name as the first input file for the MACRO command. The default file type for analysis data files is ANA. When you specify

/ANALYSIS_DATA, you can control the defaults applied to the output file specification by the placement of the qualifier in the command.

/CROSS_REFERENCE[=(function[,...])]
/NOCROSS_REFERENCE (default)

Controls whether a cross-reference listing of the locations in the source file where the specified function (or functions) is defined or referenced. If you specify more than one function, separate each with a comma and enclose the entire list in parentheses. You can specify the following functions:

ALL	Cross-references directives, macros, operation codes, registers, and symbols
DIRECTIVES	Cross-references directives
MACROS	Cross-references macros
OPCODES	Cross-references operation codes
REGISTERS	Cross-references registers
SYMBOLS	Cross-references symbols

Because the assembler writes the cross-references to the listing file, you must specify the /LIST qualifier with the /CROSS_REFERENCE qualifier. If you specify no functions in the /CROSS_REFERENCE qualifier, the assembler assumes the default value of /CROSS_REFERENCE=(MACROS,SYMBOLS). The /NOCROSS_REFERENCE qualifier excludes the cross-reference listing.

/DEBUG[=option]
/NODEBUG (default)

Includes or excludes local symbols in the symbol table or traceback information in the object module. You can replace /ENABLE and /DISABLE with /DEBUG and /NODEBUG when you use the appropriate DEBUG and TRACEBACK options. /DEBUG or /NODEBUG override debugging characteristics set with the .ENABLE or .DISABLE assembler directives. You can specify one or more of the following options:

ALL	Includes in the object module all local symbols in the symbol table, and provides all traceback information for the debugger. This qualifier is equivalent to /ENABLE=(DEBUG,TRACEBACK).
NONE	Makes local symbols and traceback information in the object module unavailable to the debugger. This qualifier is equivalent to /DISABLE=(DEBUG,TRACEBACK).
SYMBOLS	Makes all local symbols in the object module available to the debugger. Makes traceback information unavailable to the debugger. This qualifier is equivalent to /ENABLE=DEBUG and /DISABLE=TRACEBACK together.
TRACEBACK	Makes traceback information in the object module available to the debugger and local symbols unavailable to the debugger. This qualifier is equivalent to /ENABLE=TRACEBACK and /DISABLE=DEBUG together.

If you specify no options to the /DEBUG qualifier, it assumes the default value of /DEBUG=ALL.

/DIAGNOSTICS[=file-spec]
NODIAGNOSTICS (default)

Creates a file containing assembler messages and diagnostic information. If you omit the file specification, the default file name is the same as the source program; the default file type is DIA.

No wildcard characters are allowed in the file specification.

The diagnostics file is reserved for use with DIGITAL layered products, such as, but not limited to, the VAX Language-Sensitive Editor (LSE).

/DISABLE=(function[,...])
/NODISABLE

Provides initial settings for the functions disabled by the .DISABLE assembler directive. You can specify one or more of the following functions:

ABSOLUTE	Assembles relative addresses as absolute addresses
DEBUG	Includes local symbol table information in the object file for use with the debugger
TRUNCATION	Truncates floating-point numbers (if truncation is disabled, numbers are rounded)
GLOBAL	Assumes undefined symbols to be external symbols
SUPPRESSION	Suppresses listing of unreferenced symbols in the symbol table
TRACEBACK	Provides traceback information to the debugger

If you specify more than one function, separate each with a comma and enclose the list with parentheses. If you specify no functions in the /DISABLE qualifier, it assumes the default value of /DISABLE=(ABSOLUTE,DEBUG, TRUNCATION). The /NODISABLE qualifier has the same effect as not specifying the /DISABLE qualifier, or negates the effects of any /DISABLE qualifiers specified earlier on the command line.

/ENABLE=(function[,...])
/NOENABLE

Provides initial settings for the functions controlled by the .ENABLE assembler directive. The /NOENABLE qualifier has the same effect as not specifying the /ENABLE qualifier, or negates the effects of any /ENABLE qualifiers specified earlier on the command line. You can specify one or more of the functions as listed in the description of the /DISABLE qualifier, separating each with a comma and enclosing the list in parentheses. If you specify no functions in the /DISABLE qualifier, it assumes the default value of /ENABLE=(GLOBAL,TRACEBACK,SUPPRESSION).

/LIBRARY
/NOLIBRARY

Positional qualifier. The /LIBRARY qualifier cannot be used with the /UPDATE qualifier. The associated input file to the /LIBRARY qualifier must be a macro library. The default file type is MLB. The /NOLIBRARY qualifier has the same effect as not specifying the /LIBRARY qualifier,

or negates the effects of any /LIBRARY qualifiers specified earlier on the command line.

The assembler can search up to 16 libraries, one of which is always STARLET.MLB. This number applies to a particular assembly, not necessarily to a particular MACRO command. If you enter the MACRO command so that more than one source file is assembled, but the source files are assembled *separately*, you can specify up to 16 macro libraries for each separate assembly. More than one macro library in an assembly causes the libraries to be searched in reverse order of their specification.

A macro call in a source program causes the assembler to begin the following sequence of searches:

1. An initial search of the libraries specified with the .LIBRARY directive. The assembler searches these libraries in the reverse order of that in which they were declared.
2. If the macro definition is not found in any of the libraries specified with the .LIBRARY directive, a search of the libraries specified in the DCL MACRO command line (in the reverse order in which they were specified).
3. If the macro definition is not found in any of the libraries specified in the command line, a search of STARLET.MLB.

/LIST[=file-spec]
/NOLIST

Creates or omits an output listing, and optionally provides an output file specification for it. The default file type for the listing file is LIS. No wildcard characters are allowed in the file specification.

An interactive MACRO command does not produce a listing file by default. /NOLIST, present either explicitly or by default, causes errors to be reported on the current output device.

/LIST is the default for a MACRO command in a batch job. /LIST allows you to control the defaults applied to the output file specification by the placement of the qualifier in the command.

/OBJECT[=file-spec]
/NOOBJECT

Creates or omits an object module. It also defines the file specification. By default, the assembler creates an object module with the same file name as the first input file. The default file type for object files is .OBJ. No wildcard characters are allowed in the file specification.

/OBJECT controls the defaults applied to the output file specification by the placement of the qualifier in the command.

/SHOW[=(function[,...])
/NOSHOW[=(function[,...])]

Provides initial settings for the functions controlled by the assembler directives .SHOW and .NOSHOW. You can specify one or more of the following functions:

CONDITIONALS	Lists unsatisfied conditional code associated with .IF and .ENDC MACRO directives
CALLS	Lists macro calls and repeat range expansions
DEFINITIONS	Lists macro definitions
EXPANSIONS	Lists macro expansions
BINARY	Lists binary code generated by the expansion of macro calls

If you specify more than one function, separate each with a comma and enclose the list in parentheses. If you specify no functions in the /SHOW qualifier, it increments the listing level count; the /NOSHOW qualifier decrements the count in similar circumstances. Because these qualifiers contribute to the listing file, you must also specify the /LIST qualifier when you use them. Updates the input file it qualifies using the SLP batch editor and the specified update file or files. By default, the assembler assumes that the update file has the same file name as the input source file and a file type of UPD. You cannot include a wildcard character in the file specifications. If you specify more than one update file specification, separating each with a comma and enclosing the list in parentheses, the assembler merges the contents into a single list of updates before applying the updates to the source file.

The /NOUPDATE qualifier has the same effect as not specifying the /UPDATE qualifier, or negates any /UPDATE qualifiers specified earlier on the command line. The input source file and update files are not changed by the update operation. The effects of the update appear in the compiled output. If you specify the /LIST qualifier with the /UPDATE qualifier, the assembler writes an audit trail of the changes to the listing file.

example

```
$ MACRO/LIST  CYGNUS, LYRA/OBJECT=LYRAN + MYLIB/LIBRARY
```

In this example, the MACRO command requests two separate assemblies. Using .MAR as the default, MACRO assembles CYGNUS.MAR to produce CYGNUS.LIS and CYGNUS.OBJ. Then it assembles LYRA.MAR and creates a listing file named LYRA.LIS and an object module named LYRAN.OBJ. The default output file type for a listing is .LIS.

The command requests the search of the MYLIB library file in the current directory for macro definitions.

MAIL

Invokes the Mail Utility (MAIL), which is used to send messages to other users of the system. For a complete description of the Mail Utility, including more information about the MAIL command and its qualifiers, see the Reference Section.

format

MAIL *[file-spec] [recipient-name]*

MERGE

Invokes the Sort/Merge Utility to combine two through ten similarly sorted input files and create a single output file. Note that input files to be merged must be in sorted order. For a complete description of the Sort/Merge Utility, including more information about the MERGE command and its qualifiers, see the Reference Section.

format

MERGE *input-file-spec1,input-file-spec2[,...] output-file-spec*

MESSAGE

Invokes the Message Utility (MESSAGE) to compile one or more files of message definitions.

format

MESSAGE *file-spec[,...]*

MONITOR

Invokes the Monitor Utility (MONITOR) to monitor classes of systemwide performance data at a specified interval.

format

MONITOR *[class-name[,...]]*

MOUNT

Invokes the Mount Utility (MOUNT) to make a disk or magnetic tape volume available for processing. For more information about the Mount Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

NCS

Invokes the VMS National Character Set Utility to provide a convenient method of implementing alternative (non-ASCII) string collating sequences, typically using subsets of the DEC Multinational Character Set. NCS also facilitates the implementation of string conversion functions.

format

NCS *[file-spec,...]*

ON

Performs a specified action when a command or program executed within a command procedure encounters an error condition or is interrupted by CTRL/Y. The specified actions are performed only if the command interpreter is enabled for error checking or CTRL/Y interrupts (the default conditions). Use the ON command only in a command procedure.

format

ON *condition THEN [$] command*

parameters

condition

Either the severity level of an error or a CTRL/Y interrupt. Specify one of the following keywords, which may be abbreviated to one or more characters:

WARNING	Return status of warning occurs ($SEVERITY equals 0)
ERROR	Return status of error occurs ($SEVERITY equals 2)
SEVERE_ERROR	Return status of error occurs ($SEVERITY equals 4)
CONTROL_Y	CTRL/Y character occurs on SYS$INPUT

The default error condition is ON ERROR THEN EXIT.

To specify a CRTL/Y interrupt, use the following keyword:

CONTROL/Y

command
The DCL command line to be executed. You can optionally precede the command line with a dollar sign ($). If you specified an error condition as the condition parameter, the action is taken when errors equal to or greater than the specified level of error occur.

example

```
$ ON WARNING THEN EXIT
    .
    .
    .
$ SET NOON
$ RUN [SSTEST]LIBRA
$ SET ON
    .
    .
    .
```

The ON command requests that the procedure exit when any warning, error, or severe error occurs. Later, the SET NOON command disables error checking before executing the RUN command. Regardless of any status code returned by the program LIBRA.EXE, the procedure continues. The next command, SET ON, reenables error checking and reestablishes the most recent ON condition.

OPEN

Opens a file for reading, writing, or both, assigns a logical name to a file, and places the name in the process logical name table.

See the qualifier descriptions for restrictions.

format

OPEN *logical-name[:] file-spec*

parameters

logical-name[:]
Specifies the logical name and a character string to be assigned to the file.

file-spec
Specifies the name of the file or device being opened for input or output. The file type defaults to DAT. Wildcard characters are not allowed. To create a new, sequential file, specify the /WRITE qualifier.

qualifiers

/APPEND

Opens an existing file for writing and positions the record pointer at the end-of-file. New records are added to the end of the file. Use the /APPEND qualifier only to add records to an existing file. The /APPEND and the /WRITE qualifiers are mutually exclusive.

/ERROR=label

Transfers control to the location specified by the label keyword (in a command procedure) if the OPEN operation results in an error. The error routine specified for this qualifier overrides any ON condition action specified. If /ERROR is not specified, the current ON condition action is taken.

/READ (default)

Opens the file for reading. If you specify the /READ qualifier without the /WRITE qualifier, you must specify an existing file.

/SHARE[=option]

Opens the specified file as a shareable file to allow other users read or write access. If you specify /SHARE=READ, users are allowed read access to the file. If you specify /SHARE=WRITE or omit the option, users are allowed read and write access to the specified file.

/WRITE

Opens the file for writing. The following restrictions apply to the /WRITE qualifier:

- Use the /WRITE qualifier to open and create a new, sequential file.
- Use the /READ qualifier with the /WRITE qualifier to open an existing file. You cannot use OPEN/READ/WRITE to create a new file.
- The /WRITE and the /APPEND qualifiers are mutually exclusive.

example

```
$ OPEN/WRITE/ERROR=OPEN_ERROR  OUTPUT_FILE  TEMP.OUT
$ COUNT = 0
$ WRITE_LOOP:
$ COUNT = COUNT +1
$ IF COUNT .EQ. 11 THEN GOTO ENDIT
$ WRITE OUTPUT_FILE "Count is ''COUNT'."
   .
   .
   .
$ GOTO WRITE_LOOP
$ ENDIT:
$ CLOSE OUTPUT_FILE
$ EXIT
$
$ OPEN_ERROR:
$ WRITE SYS$OUTPUT "Cannot open file TEMP.OUT"
$ EXIT
```

The OPEN command with the /WRITE qualifier creates the file TEMP.OUT and assigns it the logical name OUTPUT_FILE. TEMP.OUT is a sequential file.

The /ERROR qualifier specifies that if any error occurs while opening the file, the command interpreter should transfer control to the line at the label OPEN_ERROR. The command procedure writes records to the file TEMP.OUT until the symbol COUNT equals 11.

PASSWORD

When submiting a batch job through a card reader, provides the password associated with the user name that is specified with the JOB card. Although the PASSWORD card is required, the password on the card is optional if the account has a null password.

The PASSWORD command is valid only in a batch job submitted through a card reader and requires that a dollar sign precede the PASSWORD command on the card.

format

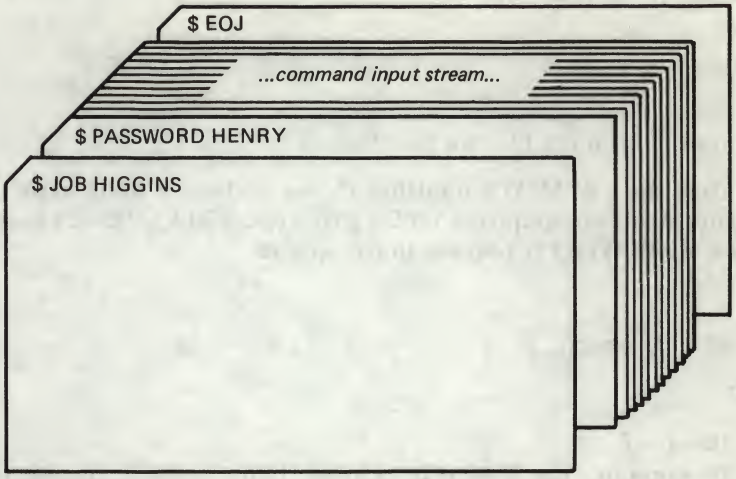
$ PASSWORD *[password]*

parameter

password

Specifies the password associated with the user name specified with the JOB command. *Password* can be 1 to 31 characters long. If you are submitting the job from an account with a null password, omit the password specifier on the PASSWORD card.

example



ZK-786-82

The JOB and PASSWORD commands precede a batch job submitted from the card reader. An EOJ command marks the end of the job.

PATCH

Invokes the Patch Utility (PATCH) to patch an executable image, shareable image, or device driver image.

format

PATCH *file-spec*

PHONE

Invokes the Phone Utility that allows you to communicate with other users on your system or any other VMS system connected to your system by DECnet–VAX.

PHONE can be used only on video terminals that are supported by the VMS screen package. These terminals include all terminals such as the VT100-, VT200-, and VT300-series that support ANSI terminal escape sequences and VT52 terminals.

format

PHONE *[phone-command]*

PRINT

Queues one or more files for printing.

Requires the /REMOTE qualifier if you include a node name in your file specification. Requires OPER privilege, EXECUTE (E) access to the queue, or WRITE (W) access to the queue.

format

PRINT *file-spec[,...]*

parameter

file-spec[,...]
Specifies one or more files to be printed. Either commas or plus signs can be used to separate file specifications. Wildcard characters are allowed . If you do not specify a file type for the first input file, the PRINT command uses the default file type LIS. Node names are allowed only when the /REMOTE qualifier is used. The file must not reside on an allocated device.

qualifiers

/AFTER=time
/NOAFTER
Holds the job until the specified time. The time can be specified as an absolute time or a combination of absolute and delta times. If the specified time has passed, the job is queued for printing immediately.

/BACKUP
/NOBACKUP
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]
/NOBEFORE
Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to

5. If a file already exists with the same file name and type as the output file, the next higher version number is used (unless the /NONEWVERSION qualifier is specified).

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each RENAME operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET> . QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])
Excludes the specified files from the RENAME operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/LOG
/NOLOG (default)
Displays the file specification of each file as it is renamed.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/NEW_VERSION (default)
/NONEW_VERSION
Assigns a new version number if an output file specification is the same as that of an existing file. The /NONEW_VERSION qualifier displays an error message if an output file specification is the same as that of an existing file. A wildcard appearing in the version field of an input or output file overrides these qualifiers.

/SINCE[=time]
Selects the RENAME operation only for those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

example

```
$ RENAME/NONEW_VERSION  SCANLINE.OBJ;2  BACKUP.OBJ
```

The RENAME command in this example renames the file SCANLINE.OBJ;2 to BACKUP.OBJ;2. The /NONEW_VERSION qualifier ensures that, if BACKUP.OBJ;2 already exists, the RENAME command does not rename the file, but instead reports the error.

REPLY

Broadcasts a message to a terminal or terminals.

See the qualifier descriptions for restrictions.

format

REPLY *["message-text"]*

parameter

message-text
Specifies the text of the message. The text must be 1 to 128 characters. Enclose the text in quotation marks ("") if it contains spaces, special characters, or lowercase characters.

qualifiers

/ABORT=identification-number
Sends a message to the user or magnetic tape file system corresponding to the unique identification number and cancels the request.

/ALL
Requires OPER privilege. Broadcasts a message to all terminals that are attached to the system or VAXcluster. These terminal must be turned on and have broadcast-message reception enabled. Incompatible with /USERNAME and /TERMINAL.

/BELL
Rings a bell at the terminal receiving a message when entered with the /ALL, /TERMINAL, or /USER qualifiers; two bells when entered with /URGENT; and three bells when entered with /SHUTDOWN.

/BLANK_TAPE=identification-number
Requires VOLPRO privilege. Sends a message to the magnetic tape file system indicated by the identification number to override the checking of volume label information. The volume label must be specified in the message text parameter. The current terminal must be enabled as an operator terminal for TAPES.

/DISABLE[=(keyword[,...])]

Requires OPER privilege. Requires OPER and SECURITY privileges for security messages. If the Operator Communication Facility (OPCOM) is running, restores to normal (that is, nonoperator) status the terminal at which the command is entered. The /DISABLE qualifier cannot be entered from a batch job. To restrict the types of messages displayed on an operator's terminal, specify one of the following keywords:

CARDS	Inhibits messages sent to the card readers
CENTRAL	Inhibits messages sent to the central system operator
CLUSTER	Inhibits messages from the connection manager pertaining to cluster state changes
DEVICES	Inhibits messages pertaining to mounting disks
DISKS	Inhibits messages pertaining to mounting and dismounting disk volumes
NETWORK	Inhibits messages pertaining to networks; the keyword CENTRAL must also be specified to inhibit network messages
OPER1 through OPER12	Inhibits messages sent to operators identified as OPER1 through OPER12
PRINTER	Inhibits messages pertaining to print requests
SECURITY	Inhibits messages pertaining to security events; requires SECURITY privilege.
TAPES	Inhibits messages pertaining to mounting and dismounting tape volumes

/ENABLE[=(keyword[,...])]

Requires OPER privilege. Requires OPER and SECURITY privileges for security messages. If the Operator Communication Facility (OPCOM) is running, designates as an operator's terminal the terminal at which the REPLY command is entered. Cannot be entered from a batch job. To enable the following types of messages displayed on an operator's terminal, specify one of the following keywords:

CARDS	Displays messages sent to the card readers
CENTRAL	Displays messages sent to the central system operator
CLUSTER	Displays messages from the connection manager pertaining to cluster state changes
DEVICES	Displays messages pertaining to mounting disks
DISKS	Displays messages pertaining to mounting and dismounting disk volumes
NETWORK	Displays messages pertaining to networks; the keyword CENTRAL must also be specified to inhibit network messages

OPER1 through OPER12	Displays messages sent to operators identified as OPER1 through OPER12
PRINTER	Displays messages pertaining to print requests
SECURITY	Allows messages pertaining to security events; requires SECURITY privilege
TAPES	Allows messages pertaining to mounting and dismounting tape volumes

/INITIALIZE_TAPE=identification-number

Sends a message to the magnetic tape file system indicated by the identification number to initialize a magnetic tape volume. This qualifier can be used whenever the file system requests the mounting of a new volume. The system performs normal protection and expiration checks before initializing the volume. The current terminal must be enabled as an operator terminal for TAPES.

/LOG
/NOLOG

Requires OPER privilege. If the Operator Communication Facility (OPCOM) is running, closes the current operator's log file and opens a new one. (The /NOLOG qualifier closes the current log file, but does not open a new log file.) The current terminal must be enabled as an operator terminal.

/NODE[=(node-name[,...])]

Sends a message to the local VAXcluster node only. The optional parameter list allows you to specify which nodes will receive the message. Default sends messages to all cluster nodes.

/NOTIFY (default)
/NONOTIFY

Sends a message describing success back to the originating terminal.

/PENDING=identification-number

Requires OPER privilege. Sends a message to the user specified by the identification number and prevents the user from entering other commands until the operator fulfills or aborts the request. The current terminal must be enabled as an operator terminal.

/SHUTDOWN

Sends a message beginning "SHUTDOWN..."; if used with /BELL, rings three bells at terminals receiving the message.

/STATUS

Requires OPER privilege. Reports the current operator status and all outstanding user requests for the terminal from which this command was entered. The current terminal must be enabled as an operator terminal.

/TEMPORARY

Designates the terminal at which the command is entered to be an operator's terminal for the current interactive session only. This qualifier is meaningful only when used with the /ENABLE qualifier.

/TERMINAL=(terminal-name[,...])

Requires OPER privilege. Broadcasts the message to specified terminals, where the terminal-name keyword is the device name of the terminal. Incompatible with /ALL and /USERNAME.

/TO=identification-number

Requires OPER privilege. Sends a message to the user or file system specified by the identification number and completes the request. The current terminal must be enabled as an operator terminal.

Note that you can also use a variation of REPLY/TO in response to a MOUNT/ASSIST command where you redirect the mount operation to another device. Whenever you must substitute a device, load the user's volume on the alternate device and ready the device before entering the REPLY command. Use the following syntax:

```
REPLY/TO=identification-number "SUBSTITUTE  device-name"
```

You can abbreviate the word SUBSTITUTE to **S** and use upper or lowercase characters. After a space, use the remainder of the message-text space to name the substituted device.

/URGENT

Sends a message beginning "URGENT..."; if used with the /BELL qualifier, rings two bells at terminals receiving the message.

/USERNAME[=(username[,...])]

Requires OPER privilege. Broadcasts a message to all terminals at which users are logged in to the system (or VAXcluster), or only to the terminals of the specified users. Overrides any NOBROADCAST settings at users' terminals.

/WAIT

Sends a message synchronously and then waits. The default is to send a message to OPCOM, which does the actual I/O. On a VAXcluster, the message is sent to the local node.

example

```
%OPCOM, 31-DEC-1988 10:19:33.21, request 5, from user SYSTEM
OPA0, Please mount OPGUIDE on DBA3:
$ REPLY/PENDING=5 "YOU'LL HAVE TO WAIT-THERE ARE SEVERAL REQUESTS BEFORE YOURS"
    .
    .
    .
REPLY/TO=5
 31-DEC-1988 10:20:25.50, request 5 completed by operator OPA0
```

In this example the OPCOM message indicates that a user wants the operator to place the disk volume labeled OPGUIDE on the disk drive DBA3 and ready the device. The REPLY/PENDING command indicates that the operator can perform the task but not immediately; the /PENDING qualifier prevents the user from entering other commands until the operator fulfills or aborts the request. After mounting the disk on the drive the operator sends a message indicating that the request has been fulfilled. When no message is specified, OPCOM sends a standard message indicating that the task has been performed.

REQUEST

Displays a message at a system operator's terminal and optionally requests a reply. All messages are logged at the operator's console and in the operator's log file, if that file is initialized.

To use this command, you must start the OPCOM process at boot time by specifying the DCL command @SYS$SYSTEM:STARTUP OPCOM in the site-specific startup command file, SYS$MANAGER:SYSTARTUP.COM.

format

REQUEST *"message-text"*

parameter

"message-text"

Specifies the text of the message to be displayed. The string can be up to 128 characters. If the string contains spaces, special characters, or lowercase characters, enclose it in quotation marks ("").

qualifiers

/REPLY

Requests a reply to the message and issues a unique identification number to which the operator sends the response. The system displays a message that the operator has been notified; you cannot enter any commands until the operator responds. If you press CTRL/C before the operator responds, you can then enter another message to the operator, or press CTRL/Z to cancel the request.

/TO=(operator[,...])

Specifies one or more operators to whom you want to send the message. Possible keywords are as follows:

CARDS	Sends the message to operators designated to respond to card reader requests
CENTRAL	Sends the message to the central system operator
CLUSTER	Sends the message to operators designated to respond to cluster-related requests
DEVICES	Sends the message to operators who mount and dismount disks
DISKS	Sends the message to operators who mount and dismount disk volumes
NETWORK	Sends the message to the network operator
OPER1 through OPER12	Sends the message to operators identified as OPER1 through OPER12
PRINTER	Sends the message to operators designated to handle print requests
SECURITY	Sends the message to operators designated to respond to security-related requests
TAPES	Sends the message to operators designated to mount and dismount tape volumes

example

```
$ REQUEST/REPLY  "Are you there?"
%OPCOM-S-OPRNOTIF, operator notified, waiting...14:54:30.33
CTRL/C
REQUEST-Enter message or cancel request with ^Z
REQUEST-Message?CTRL/Z
%OPCOM-S-OPRNOTIF, operator notified, waiting... 14:59:01.38
%OPCOM-F-RQSTCAN, request was cancelled
```

In this example the REQUEST command issues a message and requests a response. When no operator replies to the question, CTRL/C is used to interrupt the request; then CTRL/Z is used to cancel it.

RETURN

Terminates a GOSUB subroutine procedure and returns control to the command following the calling GOSUB command.

format

RETURN *[status-code]*

parameter

status-code

Defines a longword (integer) value or expression equivalent to an integer value that gives the exit status of the subroutine by defining a numeric value for the reserved global symbol $STATUS. The value can be tested by the next outer command level. The low-order three bits of the longword integer value change the value of the reserved global symbol $SEVERITY.

If you do not specify a status-code, the current value of $STATUS is saved. When control returns to the outer command level, $STATUS contains the status of the most recently executed command or program.

The low-order three bits of the status value contained in $STATUS represent the severity of the condition. The reserved global symbol $SEVERITY contains this portion of the condition code. Severity values range from zero through four, as shown in the following table:

Value	Severity
0	Warning
1	Success
2	Error
3	Information
4	Severe (fatal) error

example

```
$ SHOW TIME
  18-APR-1988 14:25:42
$ GOSUB SYMBOL
$ EXIT
$ SYMBOL:
$     SHOW SYMBOL RED
      RED = "SET DEFAULT [JONES.DCL]"
$     RETURN 1
```

The GOSUB command transfers control to the subroutine labeled SYMBOL. After the subroutine is executed, the RETURN command transfers control back to the command following the calling GOSUB statement, giving $STATUS and $SEVERITY a value of 1. The procedure then exits.

RUN (Image)

Executes an image within the context of your process. You can abbreviate the RUN command to a single letter, **R**.

If you specify an image name in the command line with an explicit version number (or a semicolon), the image runs with current process privileges. If you do not specify an explicit version number (or semicolon), the image runs with any privileges with which it was installed. If you have DECnet software installed and want to execute an image over the network, you must have READ access to the file.

format

RUN *file-spec*

parameter

file-spec
Specifies an executable image to be executed. The file type defaults to EXE. Wildcard characters are not allowed.

qualifier

/DEBUG
/NODEBUG
Executes the image under control of the debugger. The default is /DEBUG if the image is linked with /DEBUG and /NODEBUG if the image is linked without /DEBUG. The /DEBUG qualifier is invalid if the image is linked with /NOTRACEBACK. The /NODEBUG qualifier overrides the effect of LINK/DEBUG. If the image was linked with /TRACEBACK, traceback reporting is performed when an error occurs.

example

```
$ RUN LIBRA
```

The image LIBRA.EXE starts executing in the process. If the image LIBRA has been installed with amplified privileges, it runs with those privileges because you have not explicitly specified a version number or a semicolon. Alternatively, the image LIBRA.EXE still runs with its amplified privileges, if you enter the RUN command as follows:

```
$ RUN LIBRA.EXE
```

RUN (Process)

Creates a subprocess or a detached process to run an image and deletes the process when the image completes execution. A subprocess is created if any of the qualifiers except /UIC or /DETACHED is specified. A detached process is created if the /UIC qualifier is specified and you have the DETACH user privilege.

format

RUN *file-spec*

parameter

file-spec

Specifies the file name of an executable image to be executed in a separate process. The default file type is EXE. Wildcard characters are not allowed in the file specification.

qualifiers

/ACCOUNTING (default)
/NOACCOUNTING

Requires ACNT privilege to disable accounting. Logs accounting records in the system accounting file for the created process.

/AST_LIMIT=quota

Specifies the maximum number of asynchronous system traps (ASTs) that the created process can have outstanding. If you do specify an AST limit quota, the default quota established at system generation time is used. The minimum required for any process to execute is 2. The AST limit quota is nondeductible.

/AUTHORIZE
/NOAUTHORIZE (default)

Requires DETACH privilege. When the image to be executed is the system login image (LOGINOUT.EXE), this qualifier searches the user authorization file to validate a detached process. The /NOAUTHORIZE qualifier creates a detached process that runs under the control of the command interpreter.

/BUFFER_LIMIT=quota

Specifies the maximum amount of memory, in bytes, that the process can use for buffered I/O operations or for temporary mailbox creation. If you do not specify a buffered I/O quota, the default value established at system generation time is used. The minimum amount required for any process to execute is 1024 bytes.

/DELAY=delta-time

Places the created process in hibernation and awakens it after a specified time interval. If you specify both /DELAY and /INTERVAL, the first wakeup request occurs at the time specified by /DELAY. All subsequent wakeups occur at the interval specified by /INTERVAL.

/DETACHED
/NODETACHED

Creates a detached process with the same user identification code (UIC) as the current process. (To create a detached process with a different UIC, use the /UIC qualifier.) By default, the detached process has the same resource quotas as the current process; the DETACH privilege allows you to specify any quotas you need for the detached process. Unless you have the DETACH privilege, the maximum number of detached processes that you can create is limited to the quota defined by MAX_DETACH in your user authorization file.

/DUMP
/NODUMP (default)

When an image terminates because of an unhandled error, /DUMP causes the contents of the address space to be written to the file named SYS$LOGIN:IMAGEDUMP.DMP.

/ENQUEUE_LIMIT=quota

Specifies the maximum number of locks that a process can have outstanding at any one time. The default quota is that established at system generation time. The minimum required for any process to operate is 2.

/ERROR=file-spec

Defines an equivalence name string of 1 to 63 alphanumeric characters for the logical device name SYS$ERROR. The logical name and equivalence name are placed in the process logical name table for the created process. (The /ERROR qualifier is ignored if you are running SYS$SYSTEM:LOGINOUT.)

/EXTENT=quota

Specifies the maximum size to which the image being executed in the process can increase its physical memory size. The default quota is that established at system generation time. The minimum value required for any process to execute is 10 pages. The extent quota is nondeductible.

/FILE_LIMIT=quota

Specifies the maximum number of files that a process can have open at any one time. The default quota is the quota established at system generation time. The minimum amount required for any process to execute is 2. The file limit quota is pooled.

/INPUT=file-spec

Defines an equivalence name string of 1 to 63 characters for SYS$INPUT. The logical name and equivalence name are placed in the process logical name table for the created process.

/INTERVAL=delta-time

Requests that the created process be placed in hibernation and awakened at regularly scheduled intervals. If you specify the /DELAY or /SCHEDULE qualifier with the /INTERVAL qualifier, the first wakeup occurs at the time specified by /DELAY or /SCHEDULE; all subsequent wakeups occur at intervals specified by /INTERVAL. If you specify neither /DELAY nor /SCHEDULE with /INTERVAL, the first wakeup occurs immediately by default.

/IO_BUFFERED=quota

Specifies the maximum number of system-buffered I/O operations that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. The minimum required for any process to execute is 2. The buffered I/O quota is nondeductible.

/IO_DIRECT=quota

Specifies the maximum number of direct I/O operations that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. The minimum required for any

/FEED (default)
/NOFEED

Positional qualifier. Automatically inserts form feeds when pages are within 4 lines of the end of the page (line 62 on 66-line forms). You can reset the number of lines per form with the /FORM qualifier. The /[NO]FEED qualifier does not affect user-formatted files.

/FLAG[=keyword]
/NOFLAG

Positional qualifier. Controls whether a flag page is printed preceding a file. The flag page contains the name of the user submitting the job, the job entry number, and other information about the file being printed. If the /FLAG qualifier is positioned between the PRINT command and the file specifications, it can take either of two keywords:

ALL	Prints a flag page before each file in the job
ONE	Prints a flag page before the first file in the job

If you want the /FLAG qualifier to apply to individual files in a multifile job, place the qualifier directly after each file that you want to have a flag page.

/PAGE_FILE=quota

Specifies the maximum number of pages that can be allocated in the paging file for the process. The default quota is the quota established at system generation time. The minimum value required for a process to execute is 256 pages. The paging file quota is pooled.

/PRIORITY=n

Requires ALTPRI privilege to set the priority higher than your current process. Specifies the base priority at which the created process executes. The value of *n* is a decimal number from 0 through 31. The default priority is that of the current process.

/PRIVILEGES=(privilege[,...])

Requires SETPRV privilege to specify privileges that you do not have. Defines user privileges for the created process. By default, the created process has the same privileges as its creator. If you specify only one privilege, you can omit the parentheses.

For a list of process privileges, see Table 4-1 in the *VMS System Manager's Manual.*

You can also use the keyword NOSAME as the privilege parameter. If you specify /PRIVILEGES=NOSAME, the created process has no privileges.

If you specify a version number (or semicolon) in the file-spec parameter, the current process privileges are used, overriding any privileges specified with the /PRIVILEGES qualifier.

/PROCESS_NAME=process-name

Specifies a name of 1 to 15 characters for the created process. The process name is implicitly qualified by the group number of the process's user identification code (UIC). By default, the name is null.

/QUEUE_LIMIT=quota

Specifies the maximum number of timer queue entries that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. A process does not require any timer queue quota in order to execute. The timer queue entry quota is pooled.

/RESOURCE_WAIT (default)
/NORESOURCE_WAIT

Places the created process in a wait state when a resource required for a particular function is not available. If you specify /NORESOURCE_WAIT, the process receives an error status code when a resource is unavailable.

/SCHEDULE=absolute-time

Places the created process in hibernation and awakens it at the specified time.

/SERVICE_FAILURE
/NOSERVICE_FAILURE (default)
Enables or disables an exception condition notification if an error occurs during a system service request. By default, an error status code is returned to the process.

/SUBPROCESS_LIMIT=quota
Specifies the maximum number of subprocesses that the created process is allowed to create. The default quota is the quota established at system generation time. A process does not require any subprocess quota in order to execute. The subprocess limit quota is pooled.

/SWAPPING (default)
/NOSWAPPING
Requires PSWAPM privilege to inhibit process swapping. Permits the process to be swapped. By default, a process may be swapped out of the balance set whenever it is in a wait state.

/TIME_LIMIT=limit
Specifies the maximum amount of CPU time (in delta time) a created process can use. CPU time is allocated to the created process in units of 10 milliseconds. When it has exhausted its CPU time limit quota, the created process is deleted.

If this quota is not specified and the created process is a detached process, the detached process receives a default value of 0, that is, unlimited CPU time.

If this quota is not specified and the created process is a subprocess, the subprocess receives half the CPU time limit quota of the creating process.

If this quota is specified as 0, the created process has unlimited CPU time providing that the creating process also has unlimited CPU time. If, however, the creating process does not have unlimited CPU time, the created process receives half the CPU time limit quota of the creating process.

The CPU time limit quota is a consumable quota; that is, the amount of CPU time used by the created process is not returned to the creating process when the created process is deleted.

/UIC=uic
Specifies that the created process be a detached process and assigns it a user identification code (UIC).

/WORKING_SET=default

Specifies the number of pages in the working set of the created process. The default working set size is the size established at system generation time. The minimum number of pages required for a process to execute is 10 pages. The value specified cannot be greater than the quota specified with /MAXIMUM_WORKING_SET. The maximum working set quota is nondeductible.

example

```
$ RUN/INTERVAL=1:40/PROCESS_NAME=STAT   STATCHK
%RUN-S-PROC_ID, identification of created process is 00050023
    .
    .
    .
$ CANCEL STAT
```

In this example, the RUN command creates a subprocess named STAT to execute the image STATCHK.EXE. The process is scheduled to execute the image at intervals of 1 hour and 40 minutes. The process hibernates; however, because neither the /DELAY nor /SCHEDULE qualifier is specified, the first wakeup occurs immediately.

The CANCEL command subsequently cancels the wakeup requests posted by the /INTERVAL qualifier. If the process is currently executing the image, it completes the execution and hibernates.

RUNOFF

Invokes the DIGITAL Standard Runoff (DSR) text formatter to format one or more ASCII files. Creates formatted files from source DSR (RNO) files, unformatted table of contents (RNT) files, and unformatted index (RNX) files. Optionally creates intermediate (BRN) files for input to RUNOFF /CONTENTS and RUNOFF/INDEX commands.

For more information about using the commands available within DSR, see the Reference Section. For information about the DCL commands RUNOFF /CONTENTS and RUNOFF/INDEX, see the DCL command descriptions.

indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BURST[=keyword]
/NOBURST

Positional qualifier. Controls whether a burst page is printed preceding a file. (A burst page is a flag page printed over the perforation between pages for easy identification of individual files.) If the /BURST qualifier is specified between the PRINT command and the file specifications, it can take either of two keywords:

ALL	Prints a burst page before each file in the job
ONE	Prints a burst page before the first file in the job

If you want the /BURST qualifier to apply to individual files in a multifile job, place the qualifier directly after each file that you want to have a burst page. The default is /NOBURST.

/BY_OWNER[=uic]
/NOBY_OWNER

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CHARACTERISTICS=(characteristic[,...])

Specifies one or more characteristics desired for printing the files. If you specify only one characteristic, you can omit the parentheses. Use the SHOW QUEUE/CHARACTERISTICS command to see which characteristics have been defined for your system (defined with the DEFINE/CHARACTERISTIC command).

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each print operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type

a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/COPIES=n

Positional qualifier. Specifies the number of copies to print. The value of n can be from 1 to 255 and defaults to 1. If you place the /COPIES qualifier after the PRINT command name, each file in the parameter list is printed the specified number of times. If you specify /COPIES following a file specification, only that file is printed the specified number of times.

/CREATED (default)
/NOCREATED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/DELETE
/NODELETE (default)

Positional qualifier. Controls whether files are deleted after printing. If you place the /DELETE qualifier after the PRINT command name, all specified files are deleted. If you specify /DELETE after a file specification, only that file is deleted after it is printed.

/DEVICE=queue-name[:]

Places the print job in the specified queue (rather than the default queue SYS$PRINT). This qualifier is synonymous with /QUEUE, except that the /DEVICE qualifier is reserved for special use by DIGITAL. Its usage, therefore, is not recommended.

/EXCLUDE=(file-spec[,...])
/NOEXCLUDE

Excludes the specified files from the PRINT operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
/NOEXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

format

RUNOFF *file-spec[,...]*

parameter

file-spec[,...]

Specifies one or more ASCII files (containing text and DSR commands) to be formatted by the RUNOFF command. The input file type defaults to RNO; you must specify the file type for RNT and RNX files. Separate multiple files with commas. Wildcard characters are not allowed in the file specification.

DSR produces an output file having the same file name as the input file. The output file type depends on the input file type. The default output file type is MEM. Specify SYS$INPUT to type the input from your terminal or a command procedure; terminate input from the terminal by pressing CTRL/Z.

qualifiers

/BACKSPACE

Positional qualifier. Controls whether DSR uses the ASCII backspace character to perform character-by-character overprinting. By default, DSR performs line-by-line overprinting.

/BOLD[=n]
/NOBOLD

Positional qualifier. Specifies the number of times characters are overstruck in a bolding operation. You can specify the number of times DSR overprints flagged text by stating a value for n. N must be 0 or a positive integer and defaults to 1. A specification of /BOLD=0 or /NOBOLD disables all bolding, even if the appropriate flags are recognized and enabled.

/CHANGE_BARS[="character"]
/NOCHANGE_BARS

Positional qualifier. Controls whether DSR generates change bars in the formatted file. The default change-bar character is the vertical bar (|). The change bars appear 3 spaces to the left of the lines of text that you have marked for change bars. You can replace the default change-bar character by supplying a substitute character for the /CHANGE_BARS[="character"] qualifier. You must specify the replacement character as either a character enclosed in quotation marks or as an octal, decimal, or hexadecimal value for the desired character.

/DEBUG[=(option[,...])]
/NODEBUG (default)

Positional qualifier. Traces certain operations by placing the DSR commands in the output file. The options are as follows:

ALL	Specifies all five options (CONDITIONALS, CONTENTS, FILES, INDEX, and SAVE_RESTORE).
CONDITIONALS	Causes DSR to ignore all conditional processing commands (.IF, .IFNOT, .ELSE, .ENDIF) in the input file. DSR includes both "true" and "false" conditional information in the output file along with formatted text.
CONTENTS	Causes DSR to output all .SEND TOC commands along with the text being sent to the table of contents.
FILES	Causes DSR to output all .REQUIRE commands as well as the text of the require files.
INDEX	Causes DSR to output the indexing commands, .INDEX and .ENTRY, in addition to the text to which they refer.
SAVE_RESTORE	Causes DSR to output all .SAVE and .RESTORE commands.

If you specify more than one option, separate them with commas and enclose the list in parentheses. If you specify /DEBUG without specifying any options, ALL is assumed.

/DEVICE=(option[,...])

Positional qualifier. Controls whether DSR generates an output file (LNI) that is suitable for printing on an LN01, LN01E, or an LN03 laser printer. You can choose options from the following list to indicate output device, page orientation, and type of emphasis for flagged characters in your LNI file:

LN01	Produces an output file suitable for printing on an LN01 laser printer; the default paper size is 8 1/2 by 11 inches; the default mode is PORTRAIT. The output file name is the same as the input file name; the default file type is LNI.
LN01E	Produces an output file suitable for printing on an LN01E laser printer using the standard European paper size (A4). The output file name is the same as the input file name. The default file type is LNI; the default mode is PORTRAIT. Incompatible with LN01.
LN03	Produces an output file suitable for printing on an LN03 laser printer; the default paper size is 8 1/2 x 11 inches. The output file name is the same as the input file name. The default file type is LNI; the default mode is PORTRAIT.
LANDSCAPE	Causes the appropriate fonts for landscape mode to be loaded into the LN01; pages are printed with the long dimension at top using a smaller type size. (The page is 11 inches wide and 8 1/2 inches long.) Allowable page dimensions are 0 to 73 lines per page and 0 to 132 characters per line. Incompatible with PORTRAIT.
PORTRAIT (default)	Causes the appropriate fonts for portrait mode to be loaded into the LN01; pages are printed with the short dimension at top using a larger type size. (The page is 8 1/2 inches wide and 11 inches long.) Allowable page dimensions are 0 to 66 lines per page and 0 to 80 characters per line. Incompatible with LANDSCAPE.

ITALIC (default)	Causes the italic and bold-italic fonts to be loaded into the LN01 printer. Italicizes characters flagged for underlining. Italicized characters can also be bolded depending on the type of emphasis you specify in your input file.
UNDERLINE	Causes the text and bold fonts to be loaded into the LN01. Underlines characters flagged for underlining. The LN01 allows only 63 consecutive characters (counting a space as a character) to be underlined per line. If you want to underline individual words and not the spaces between them, you will be able to underline only 63 words per line. Incompatible with ITALIC.

/DOWN[=n]
/NODOWN (default)

Positional qualifier. Controls whether DSR inserts a specified number of blank lines at the top of each page. These blank lines precede any header information. The number of blank lines you specify (n) does not affect the number of text lines on a page.

If you specify the /DOWN qualifier without a value, five blank lines are inserted. If you specify /DOWN=0 or omit the qualifier, no blank lines are inserted, except those associated with the print device or header layout.

/FORM_SIZE=n

Specifies the maximum number of lines per page including running heads and running feet. Defaults to /FORM_SIZE=66, which is standard for 11-inch paper. For laser printers, set the number of lines as follows:

Paper Size	Lines	Mode
8.05	69	Landscape
8.28	71	Landscape (LN01E default)
8.51	73	Landscape (LN01, LN03 default)
11.00	66	Portrait (LN01, LN03 default)
11.66	70	Portrait (LN01E default)
12.33	74	Portrait
13.00	78	Portrait
14.00	84	Portrait

/INTERMEDIATE[=file-spec]
/NOINTERMEDIATE (default)

Positional qualifier. Controls whether DSR generates an intermediate output file that can be used as input to the DSR table of contents utility and the DSR indexing utility. See the descriptions of RUNOFF/CONTENTS and RUNOFF/INDEX in the Reference Section for more information on producing tables of contents and indexes.

/LOG
/NOLOG (default)

Controls whether a termination message is displayed at the terminal after successful completion of the DSR operation. The message states the DSR version number, the number of diagnostic messages (if any), the number of output pages, and the output file specification.

/MESSAGES=(option[,...])

Positional qualifier. Specifies the destination of all DSR error messages. To indicate a specific destination, use one or both of the following options:

OUTPUT	Messages are sent to the output MEM file
USER	Messages are displayed on the terminal (SYS$ERROR)

If you specify both options, separate them with commas and enclose the list in parentheses. The default, /MESSAGES=(OUTPUT,USER), sends messages to the output MEM file and displays them on the terminal.

/OUTPUT[=file-spec]
/NOOUTPUT

Positional qualifier. Specifies that an output file is to be produced and optionally names it. If you specify /OUTPUT without a file specification, or if you omit the qualifier, the directory and file name default to that of the DSR file. If you specify /NOOUTPUT, no output file is produced. The output file type depends on the input file type. The default input file type is RNO and the default output file type is MEM.

The file type defaults to one of the following:

BLB	For an RNB input file
CCO	For an RNC input file
DOC	For an RND input file
ERR	For an RNE input file
HLP	For an RNH input file
LNI	For an RNO input file with /DEVICE set to LN01, LN01E, or LN03
MAN	For an RNM input file
MEC	For an RNT input file
MEM	For an RNO input file with no /DEVICE specification
MEX	For an RNX input file
OPR	For an RNP input file
PLM	For an RNL input file
STD	For an RNS input file

/PAGES=string

Positional qualifier. Specifies that only the pages within the specified range be generated as output. By default, DSR generates output for all pages. Specify the range as follows:

```
start-page-no:end-page-no,...
```

You can specify up to five ranges (/PAGES="2-9:2-12, 4-1:4-10, 5-9:5-9, A-1:A-3, Index-1:Index-5"). You can omit the colon and the end page number on the last range. You can omit the quotation marks if you specify only one range. Page numbers must be specified in their default form, not the form specified in a .DISPLAY command. You can specify just the appendix letter or name to produce an entire appendix. You can specify just the word INDEX to produce an entire index.

/PAUSE
/NOPAUSE (default)

Controls whether DSR pauses after printing each page of output. You can use the /PAUSE qualifier to insert single sheets of paper or reproduction masters into hardcopy output devices. When output is halted, the terminal bell rings to remind you to insert a new form. Press the space bar to resume processing. Do not use this qualifier in a batch job.

/REVERSE_EMPHASIS

Positional qualifier. Directs DSR to change the order in which flagged text is underlined on an output device. If you use this qualifier, the printer first prints the characters to be underlined, then issues a carriage return without a line feed, and prints the underscores to underline the flagged text. If you view your file on the terminal, the flagged text is overwritten by the underline character.

/RIGHT[=n]
/NORIGHT (default except for LN01)

Positional qualifier. Causes the text on each page (including header information) to be shifted to the right the number of columns specified by n. This qualifier does not affect the page width. If you specify /RIGHT without specifying a number, text is shifted to the right five spaces. If you specify a value of zero or omit the qualifier, no shift occurs.

The defaults (if /RIGHT is not specified) for LN01 files are as follows:

Mode	LN01	LN01E	LN03
Landscape	9	13	9
Portrait	2	2	2

/SEPARATE_UNDERLINE[="character"]

Positional qualifier. Prints underlines as separate characters on the next line instead of overstriking with underscores on the same line. The value specifies

the character to be used for the underline character and defaults to a hyphen (-). You can specify the underline character as a single printable character or a number preceded by a radix indicator (%D, %O, or %X) to represent the ASCII value of a printable or nonprintable character. Incompatible with the /[NO]UNDERLINE_CHARACTER qualifiers.

/SEQUENCE
/NOSEQUENCE (default)
Positional qualifier. Controls whether DSR precedes the lines in the output file with the line numbers of the corresponding lines in the DSR file. For editors that generate line numbers in the input file, the /SEQUENCE qualifier causes similar numbering to appear in the output file. The line numbers appear in the left margin at the beginning of each line of output. If the text editor does not generate sequential numbers in the input file, sequential numbers are still generated in the output file, but without leading zeros.

/SIMULATE
/NOSIMULATE (default)
Controls whether DSR uses line feeds or form feeds to advance to the top of each page. For devices that do not have a form-feed capability, use /SIMULATE to generate enough blank lines to cause a skip to the top of each new page. The /SIMULATE qualifier also causes a pause before the first page of output. To continue after the pause, press the space bar.

/UNDERLINE_CHARACTER[="character"]
/NOUNDERLINE_CHARACTER
Positional qualifier. Specifies the character to be used for the underline character. Defaults to an underscore (_). You can specify the underline character as a single printable character (enclosed in quotation marks) or as a number preceded by a radix indicator (%D, %O, or %X) to represent the ASCII value of a printable or nonprintable character. A specification of /NOUNDERLINE_CHARACTER overrides any .ENABLE UNDERLINING command in the DSR file. Incompatible with /SEPARATE_UNDERLINE.

/VARIANT=string
Positional qualifier. Controls the processing of the conditional commands (.IF, .IFNOT, .ELSE, and .ENDIF) by specifying the names of the segments to be processed. You must name conditional structures introduced by .IF to process them. You must name conditional structures introduced by .IFNOT to exclude them. You must not name conditional structures introduced by .ELSE to process them. If you specify multiple names in a string, separate them by commas and enclose the string in quotation marks.

example

```
$ RUNOFF CHAPT1/RIGHT=10,CHAPT2
```

The RUNOFF command in this example produces a CHAPT1.MEM file with margins ten spaces to the right of the margins specified in the input file CHAPT1.RNO. It also generates a CHAPT2.MEM file whose margins are not affected by the /RIGHT=10 qualifier.

RUNOFF/CONTENTS

Invokes the DIGITAL Standard Runoff (DSR) table of contents utility to create an RNT file that can be processed by DSR to make a table of contents. The input file for this command is an intermediate binary file (BRN) that is produced with the RUNOFF command and the /INTERMEDIATE qualifier (see the RUNOFF command).

For more information about using the DSR table of contents utility and the commands available within DSR, see the Reference Section.

format

RUNOFF/CONTENTS *file-spec[,...] or file-spec[+...]*

parameter

file-spec[,...] or file-spec[+...]
Specifies one or more intermediate binary files (BRN) that contain information (chapter titles, header levels, sections, and so on) for making a table of contents. To create a BRN file, use the RUNOFF command with the /INTERMEDIATE qualifier. If you omit the input file type, the DSR table of contents utility uses a default file type of BRN. RUNOFF/CONTENTS will also process BTC files that the previous version of DSR produced. For single input files, the table of contents utility produces an output file with the same file name as the input file. The output file type is RNT.

If you separate multiple input files with commas, separate RNT files for each input file are created. If you separate multiple input files with plus signs (+), a single RNT file that contains table of contents information for all of the input files is created. The default output file name is the same as the first input file name; the default file type is RNT. Wildcard characters are not allowed in the file specification.

qualifiers

/BOLD
/NOBOLD (default)
Controls whether the bolding specified in chapter and header titles in the input file appears in the table of contents.

/DEEPEST_HEADER=n
Controls how many levels of header levels are output in the table of contents. You can specify any number of header levels (up to 6) to be displayed by changing the value of n. The default is /DEEPEST_HEADER=6.

/IDENTIFICATION
/NOIDENTIFICATION (default)
Controls whether the current version number of the DSR table of contents utility is reported.

/INDENT
/NOINDENT (default)
Controls how many spaces the header levels after level 1 are indented in the table of contents. If you omit this qualifier, or if you specify /NOINDENT, all header levels after header level 1 are indented 2 spaces. If you specify /INDENT, each header level after header level 1 is indented 2 spaces beyond the preceding header level.

/LOG
/NOLOG (default)
Controls whether the DSR table of contents utility displays the name of each input file as it is processed and after it is processed. The name of each output file created may also be displayed. If there are any errors in processing, the DSR table of contents utility sends messages to the terminal even if /NOLOG is in effect.

/OUTPUT[=file-spec]
/NOOUTPUT
Specifies that an output file is to be produced and optionally names it. If you specify the /OUTPUT qualifier without a file specification, or if you omit the qualifier entirely, the output file name matches the input file name. The default file type is RNT. The /NOOUTPUT qualifier suppresses the creation of an output file. You can use /NOOUTPUT to check an input file for errors without using system resources to generate an output file.

/PAGE_NUMBERS=(option[,...])
Controls whether the page number references in the table of contents are running page numbers or chapter-oriented page numbers; also controls how many levels of headers have page references listed in the table of contents. To specify these options, select from the following list:

Option	Purpose
LEVEL=n	Specifies that header levels up to and including header level n have page numbers listed in the table of contents. The default is to display page numbers for 6 levels of headers.
NORUNNING	Specifies chapter-oriented page numbers (such as 1-3, 10-42). You can specify chapter-oriented numbers for the table of contents even if the document does not have chapter-oriented numbers. NORUNNING is the default.
RUNNING	Specifies running page numbers (such as 3, 42). You can specify running page numbers for the table of contents even if the document does not have running page numbers.

If you supply more than one option, separate them with commas and enclose the list in parentheses.

/REQUIRE=file-spec
/NOREQUIRE (default)

Allows you to change or delete the heading on the first page of a table of contents. The default heading is the word CONTENTS centered on the page and followed by one blank line. You can either substitute another word as a heading, or have no heading.

To change the heading, do one of the following:

- If you do not want any heading, specify a null file as the file specification for /REQUIRE.

  ```
  $ RUNOFF/CONTENTS/REQUIRE=nl:
  ```

- If you want to use a different heading, create or edit a file that specifies the heading that you want. Use the file that you create as the file specification for the /REQUIRE qualifier.

/SECTION_NUMBERS (default)
/NOSECTION_NUMBERS

Controls whether the DSR table of contents utility displays section numbers in the table of contents. The /SECTION_NUMBERS qualifier displays section numbers for all header levels in the table of contents. /NOSECTION_NUMBERS suppresses the display of section numbers for all header levels.

/UNDERLINE
/NOUNDERLINE (default)

Controls whether the underlining specified in chapter and header titles in the input file appears in the table of contents.

example

```
$ RUNOFF/INTERMEDIATE CHPT1,CHPT2,CHPT3
```

Before using RUNOFF/CONTENTS, you must use RUNOFF/INTERMEDIATE to create a BRN file as input for the DSR table of contents utility. The command line in this example creates three separate files: CHPT1.BRN, CHPT2.BRN, and CHPT3.BRN.

RUNOFF/INDEX

Invokes the DIGITAL Standard Runoff (DSR) indexing utility to create an RNX file that can be processed by DSR to create an index. The input file for this command is an intermediate binary file (BRN) that is produced with the RUNOFF command and the /INTERMEDIATE qualifier (see the RUNOFF command). For more information about using the DSR indexing utility and the commands available within DSR, see the Reference Section.

format

RUNOFF/INDEX *file-spec[,...] or file-spec[+...]*

parameter

file-spec[,...] or file-spec[+...]

Specifies one or more intermediate binary files (BRN) that contain information (index entries, page number references, and so on) for making an index. To create a BRN file, use the RUNOFF command with the /INTERMEDIATE qualifier. See the RUNOFF command for more information on the /INTERMEDIATE qualifier.

If you omit the input file type, the DSR indexing utility uses a default file type of BRN. RUNOFF/INDEX also processes BIX files that the previous version of DSR produced. For single input files, the indexing utility produces an output file with the same file name as the input file. The output file type is RNX. If you separate multiple input files with commas, separate RNX files for each input file are created. If you separate multiple input files with plus signs (+), a single RNX file that contains indexing information for all of the input files is created. The default output file name is the same as the first input file name; the default file type is RNX. Wildcard characters are not allowed in the file specification.

qualifiers

/IDENTIFICATION
/NOIDENTIFICATION (default)

Reports the current version number of the DSR indexing utility.

/LINES_PER_PAGE=n

The value n specifies the number of lines of index entries on each page of the finished index. This number does not include the number of lines required for headings and footings. The default is 55 lines. This value is designed to work properly in the default formatting environment of DSR. You must calculate the value n if you change the default environment in any of the following ways:

- If you use subtitles in the document that requires the RNX file
- If you make the page length for the document anything other than 58 lines per page
- If you use any .LAYOUT other than zero (0)

To calculate the correct value for /LINES_PER_PAGE use the following formula:

```
/LINES_PER_PAGE=n
n = .PAGE SIZE ( the first parameter is length value)
    minus 4 if subtitles are used, minus 3 if no subtitles
    minus the number of lines reserved for .LAYOUT 1,
          .LAYOUT 2, or .LAYOUT 3.
```

/LOG
/NOLOG (default)

Controls whether the DSR index utility displays the name of each input file as it is processed and after it is processed, as well as the name of each output file created. If there are any errors in processing, INDEX sends messages to the terminal even if /NOLOG is in effect.

/OUTPUT[=file-spec]
/NOOUTPUT

Specifies that an output file is to be produced and optionally names it. If you specify the /OUTPUT qualifier without a file specification, or if you omit the qualifier entirely, the output file name matches the input file name. The default file type is RNX. You can change the name of the output file by supplying a file specification for the value file-spec. The /NOOUTPUT qualifier suppresses the creation of an output file. You can use /NOOUTPUT to check an input file for errors without using system resources to generate an output file.

/PAGE_NUMBERS=option

Controls whether the page number references in the index are running page numbers or chapter-oriented page numbers. To specify the type of page numbers you want, select from the following options:

Option	Purpose
NORUNNING	Specifies chapter-oriented page numbers (such as 1-3, 10-42). You can specify chapter-oriented numbers for an index even if they do not appear in the document. NORUNNING is the default.
RUNNING	Specifies running page numbers (such as 1, 50, 230). You can specify running page numbers for an index even if the document does not display running page numbers.

/REQUIRE=file-spec
/NOREQUIRE (default)

Allows you to change the heading on the first page of an index. The default heading is the word INDEX centered on the page and followed by three blank lines. The substitute heading is contained in the file you specify, which can contain DSR commands and text.

To change the heading:

1. Create or edit a file that specifies the format and the text that you want as the heading on the first index page.

2. Use the file you create as the file-spec for /REQUIRE.

See the /RESERVE qualifier for more information.

/RESERVE=n
/NORESERVE (default)
Allows you to reserve space at the top of the first page of the index for text or white space that you want to include with the /REQUIRE=file-spec qualifier. Determine how many lines of text or white space you are adding to the top of the first page of the index. Use this number as the value n for the /RESERVE qualifier.

example

```
$ RUNOFF/INDEX/LINE_PER_PAGE=52 CHPT2
```

In this example, the RUNOFF/INDEX command takes the file CHPT2.BRN as input and creates CHPT2.RNX. The RNX file produces an index with 52 lines of index entries per page. The lines per page had to be adjusted because the writer used a page layout with the page numbers centered at the bottom of the page (.LAYOUT 1, .LAYOUT 2, .LAYOUT 3). This page layout takes up three more spaces than .LAYOUT 0, which is the default for DSR. To produce the final index, you must use the RNX file as input to DSR.

SEARCH

Searches one or more files for the specified string(s) and displays those lines containing that string or strings.

format

SEARCH *file-spec[,...] search-string[,...]*

parameters

file-spec[,...]
Specifies one or more files to be searched. You must specify at least one file name. If you specify two or more file names, separate them with commas. Wildcard characters are allowed in the file specification.

search-string[,...]
Specifies the character string to be located in the specified files. Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each SEARCH operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET> . QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXACT
/NOEXACT (default)

Controls whether the SEARCH command matches the search string exactly or treats uppercase and lowercase letters as equivalents. By default, SEARCH ignores case differences in letters.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the SEARCH operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/FORMAT=option

Formats output in one of five ways:

DUMP	Displays all control characters (including <HT> , <CR> , and <LF>) and nonprintable characters as ANSI mnemonics.
NONULLS	Same as DUMP, but removes all null characters from the input file before reformatting.
NOFF	Replaces control characters in text with ANSI mnemonics (for example, CTRL/C is replaced with <ETX>). The terminal formatting characters <HT> , <CR> , <LF> , <VT> are passed without change. Form feed characters are replaced with <FF> .
PASSALL	Moves control and nonprintable characters to the output device without translating them. The terminal driver cannot send 8-bit characters to the terminal unless either SET TERMINAL/PASSALL or SET TERMINAL/EIGHT_BIT is already in effect.
TEXT	Replaces control characters in text with ANSI mnemonics (for example, CTRL/C is replaced with <ETX>). The terminal formatting characters <HT> , <CR> , <LF> , <VT> , and <FF> are passed without change. TEXT is the default format.

/HEADING (default)
/NOHEADING

Includes file names in the output file and displays a line of 30 asterisks as a window separator between groups of lines that belong to different files. With the default heading format, file names are printed only when more than one file is specified or when wildcard characters are used.

/LOG
/NOLOG (default)
Outputs a message to the current SYS$OUTPUT device for each file searched. The message includes the file name, the number of records, and the number of matches for each file searched.

/MATCH=option
Interprets and matches multiple search strings in one of the following ways:

AND	A match occurs only if the record contains all the strings.
NOR	A match occurs only if the record contains none of the strings.
NAND	A match occurs only if the record does not contain all of the strings.
OR	A match occurs if the record contains any of the strings.

/MODIFIED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/NUMBERS
/NONUMBERS (default)
Controls whether the source line number is displayed at the left margin of each line in the output.

/OUTPUT[=file-spec]
/NOOUTPUT
Controls whether the results of the search are output to a specified file. The output is sent to the current default output device (SYS$OUTPUT) if you omit the /OUTPUT qualifier or omit the file specification with the qualifier.

/REMAINING
/NOREMAINING (default)
Includes in the output all records from the first matched record to the end of the file. This qualifier overrides the value n in /WINDOW, but allows /WINDOW=n1.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/STATISTICS
/NOSTATISTICS (default)

Controls whether the following statistics about the search are displayed:

- Number of files searched
- Number of records searched
- Number of characters searched
- Number of records matched
- Number of lines printed
- Buffered I/O count
- Direct I/O count
- Number of page faults
- Elapsed CPU time
- Elapsed time

/WINDOW[=(n1,n2)]
/NOWINDOW (default)

Specifies the number of lines to be displayed with the search string. If you specify the /WINDOW qualifier without the value n1 and n2, two lines above the search string, the search string, and the two lines below the search string are included in the output. If you specify /WINDOW with a single number (n1), n1 specifies the number of lines to display including the search string. Half the lines precede the matched search string and half follow it. (If n is even, 1 line is added to the lines following the matched search string.) If you specify n1 and n2, the /WINDOW qualifier displays n1 lines above the search string, the search string, and n2 lines below the search string. Either of these numbers can be zero. If you specify /WINDOW=0, the file name of each file containing a match (but no records) is included in the output. If you omit the /WINDOW qualifier, only the line containing a match is displayed.

example

```
$ SEARCH/OUTPUT=RESULTS.DAT/WINDOW=9 DISLIST.MEM NAME
```

The SEARCH command searches the file DISLIST.MEM for occurrences of the character string NAME and sends the output to the file RESULTS.DAT. The four lines preceding and following each occurrence of NAME are included in the output.

SET ACCOUNTING

Enables or disables the logging of various activities in the accounting log file SYS$MANAGER:ACCOUNTNG.DAT. You can also use SET ACCOUNTING to close the current accounting log file and open a new one with a version number incremented by 1.

Requires OPER privilege.

format

SET ACCOUNTING

parameters

None.

qualifiers

/DISABLE[=(keyword[,...])]

Disables the logging of all activities in the accounting log file. To disable specific activities, you include one or more keywords with /DISABLE. You can specify the following keywords:

Keyword	Function
BATCH	Inhibits/allows the recording of batch job termination
DETACHED	Inhibits/allows the recording of detached process termination
IMAGE	Inhibits/allows the recording of image activation
INTERACTIVE	Inhibits/allows the recording of interactive job termination
LOGIN_FAILURE	Inhibits/allows the recording of login failures
MESSAGE	Inhibits/allows the recording of user messages
NETWORK	Inhibits/allows the recording of network job termination
PRINT	Inhibits/allows the recording of all print jobs
PROCESS	Inhibits/allows the recording of all process termination
SUBPROCESS	Inhibits/allows the recording of all subprocess termination

/ENABLE[=(keyword[,...])]

Enables the logging of all activities in the accounting file. To enable specific activities, you include one or more keywords with /ENABLE. Use the same keywords with /ENABLE that you use with /DISABLE.

/NEW_FILE

Closes the current accounting file and opens a new version of that file.

example

```
$ SET ACCOUNTING/ENABLE=(BATCH,INTERACTIVE)
```

The command in this example requests that all batch and interactive jobs be recorded in the accounting file at job termination.

SET ACL

Allows you to create or modify the access control list (ACL) of an object. Alternatively, you may use the VMS Access Control List (ACL) Editor to manipulate ACLs.

format

SET ACL *object-name*

parameter

object-name
Specifies the object whose access control list (ACL) is being modified. Wildcard characters are allowed in object names only if the object type is FILE. Each file must be a disk file on a Files–11 Structure Level 2 formatted volume. Logical name tables must be system logical name tables.

qualifiers

/ACL[=(ace[,...])]
Specifies one or more access control entries (ACEs) to be modified. When no ACE is specified, the entire access control list is affected. Separate multiple ACEs with commas. The specified ACEs are inserted at the top of the ACL unless the /AFTER qualifier is given. (Note that security alarm ACEs are always placed at the beginning of the ACL.)

/AFTER=ace
Indicates that all access control entries (ACEs) specified with the /ACL qualifier will be added after the ACE specified with the /AFTER qualifier. By default, any ACEs added to the ACL are always placed at the top of the list. (Note that security alarm ACEs are always placed at the beginning of the ACL.)

/BEFORE[=time]
Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/BY_OWNER[=uic]
Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Issues a request for confirmation before each modification. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED

Modifies the ACLs of files selected according to their creation date. Relevant only with the /BEFORE and /SINCE qualifiers.

/DEFAULT

Creates an ACL for the specified files as if the files were newly created. For a directory file, the /DEFAULT qualifier propagates the entire ACL (except ACEs with the NOPROPAGATE option) so that a particular access protection can be propagated throughout a directory tree. For all other files, the /DEFAULT qualifier propagates the DEFAULT option ACEs in the ACL of the parent directory to the ACL of the specified files.

/DELETE

Indicates that the access control entries (ACEs) specified with the /ACL qualifier are to be deleted. If no ACEs are specified with /ACL, the entire ACL is deleted (except those with the PROTECTED option).

/EDIT

Invokes the ACL Editor and allows you to use the /JOURNAL, /MODE, or /RECOVER qualifiers. Any other qualifiers specified with /EDIT are ignored.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the SET ACL operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/JOURNAL[=file-spec]
/NOJOURNAL
Controls whether a journal file is created from the editing session. By default, a journal file is created if the editing session ends abnormally. You must specify /EDIT in order to use this qualifier.

/LIKE=(OBJECT_TYPE=type,OBJECT_NAME=name)
Deletes the ACL of the specified object and replaces it with the ACL of the object specified with /LIKE.

You can specify the following keywords for OBJECT_TYPE: DEVICE, FILE, SYSTEM_GLOBAL_SECTION, GROUP_GLOBAL_SECTION, QUEUE, or LOGICAL_NAME_TABLE. This qualifier cannot be used with the /EDIT qualifier.

/LOG
/NOLOG (default)
Controls whether the SET ACL command displays the object name of the object that has been affected by the command. This qualifier cannot be used with the /EDIT qualifier.

/MODE=[NO]PROMPT
Determines whether the ACL editor prompts for field values. By default, the ACL editor selects prompt mode. You must specify the /EDIT qualifier to use this qualifier.

/NEW
Indicates that any existing ACE in the ACL of the object specified with SET ACL (except those with the PROTECTED option) is to be deleted. To use the /NEW qualifier, you must specify a new ACL or ACE with the /ACL, /LIKE, or /REPLACE qualifier. This qualifier cannot be used with the /EDIT qualifier.

/OBJECT_TYPE=type
Specifies the type of the object whose ACL is being edited. By default, the ACL editor assumes that the object whose ACL is being edited is a file. If the object is not a file, the /OBJECT qualifier is required. Possible keywords are as follows: FILE (includes directory files), DEVICE, SYSTEM_GLOBAL_SECTION, GROUP_GLOBAL_SECTION, QUEUE, or LOGICAL_NAME_TABLE.

/RECOVER[=file-spec]
/NORECOVER (default)
Specifies the name of the journal file to be used in a recovery operation. If the file specification is omitted with /RECOVER, the journal is assumed to have the same name as the input file and a file type of JOU. No wildcard characters are allowed with the /RECOVER file-spec parameter. You must specify /EDIT in order to use this qualifier.

/REPLACE=(ace[,...])

Deletes the access control entries (ACEs) specified with the /ACL qualifier and replaces them with those specified with /REPLACE. Any ACEs specified with the /ACL qualifier must exist and must be specified in the order in which they appear in the ACL. This qualifier cannot be used with the /EDIT qualifier.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

example

```
$ SET ACL/LIKE=(OBJECT_TYPE=FILE,OBJECT_NAME=USER.LIS) ACCOUNTS.LIS
```

This example replaces the ACL of the file ACCOUNTS.LIS with the ACL for the file USER.LIS.

SET AUDIT

Enables or disables security auditing on a VMS system. The SET AUDIT command can also be used to specify the auditing failure mode on the system. (Note that you must specify the /ALARM qualifier when enabling or disabling security auditing.)

Requires the SECURITY privilege.

format

SET AUDIT

qualifiers

/ALARM

Causes alarm messages to be sent to all terminals enabled as security operators. See the description of the DCL command REPLY/ENABLE for details on how to enable terminals as security operators. The /ALARM qualifier is required when enabling or disabling security either /ENABLE or /DISABLE are required.

/DISABLE=(keyword[,...])

Disables security auditing for the specified events. To disable alarms for all events, specify the keyword ALL. You can also specify the appropriate keywords to selectively disable alarms for from one to all events that are currently enabled. You must specify at least one keyword. See the /ENABLE qualifier description for a list of the keywords to use with the /DISABLE qualifier.

/ENABLE=(keyword[,...])

Enables security auditing for the specified events. To enable alarms for all events, specify the keyword ALL. You can also specify the appropriate keywords to selectively enable alarms for from one to all events that are currently enabled. You must specify at least one keyword.

The possible events that may be specified in the keyword list of either the /ENABLE or /DISABLE qualifier are as follows:

ACL	An event requested by an access control list (ACL) item, including ACLs on files and global sections.
ALL	All possible events.
AUDIT	An event resulting from the execution of a SET AUDIT command.
AUTHORIZATION	The modification of any portion of the system user authorization file (SYSUAF) or network proxy authorization file (NETPROXY), including any password changes; the modification of any portion of the rights database.
BREAKIN=(keyword[,...])	The occurrence of one or more of the following classes of break-in attempts, as specified by one or more of the keywords:

ALL	All possible sources of break-ins, as defined by the remaining keywords
DETACHED	Detached process break-in attempt
DIALUP	Dialup break-in attempt
LOCAL	Local break-in attempt
NETWORK	Network server break-in attempt
REMOTE	Remote break-in attempt

FILE_ACCESS=(keyword[,...])

The occurrence of file and global section access events (regardless of the value specified in the file's access control list, if any). You can specify one or more of the following keywords to describe the file access event to be noted.

ALL	All types of file access events, as defined by the remaining keywords.
BYPASS [:access [,access...]]	Successful file access due to the use of the BYPASS privilege
FAILURE [:access [,access...]]	Unsuccessful file access
GRPPRV [:access [,access...]]	Successful file access due to the use of the GRPPRV privilege
READALL [:access [,access...]]	Successful file access due to the use of the READALL privilege
SUCCESS [:access [,access...]]	Successful file access
SYSPRV [:access [,access...]]	Successful file access due to the use of the SYSPRV privilege

Most of the keywords permit you to define the type of file access that was obtained with the following keywords:

ALL	All types of file access events, as defined by the remaining keywords. If no access types are specified, ALL is assumed by the system.
READ	READ access
WRITE	WRITE access
EXECUTE	EXECUTE access
DELETE	DELETE access
CONTROL	Owner access

INSTALL

The occurrence of any INSTALL operations.

LOGFAILURE=(keyword[,...])

The occurrence of one or more of the following classes of login failure, as specified by one or more of the keywords:

ALL	All possible types of login failures, as defined by the remaining keywords
BATCH	Batch process login failure
DETACHED	Detached process login failure
DIALUP	Dialup interactive login failure
LOCAL	Local interactive login failure
NETWORK	Network server task login failure
REMOTE	Interactive login failure from another network node, for example, with a SET HOST command
SUBPROCESS	Subprocess login failure

LOGIN=(keyword[,...])

The occurrence of one or more of the following classes of login attempts, as specified by one or more of the keywords:

ALL	All possible sources of logins, as defined by the remaining keywords
BATCH	Batch process login
DETACHED	Detached process login
DIALUP	Dialup interactive login
LOCAL	Local interactive login
NETWORK	Network server task login
REMOTE	Interactive login from another network node, for example, with a SET HOST command
SUBPROCESS	Subprocess login

LOGOUT=(keyword[,...])	The occurrence of one or more of the following classes of logouts, as specified by one or more of the keywords:	
	ALL	All possible sources of logouts, as defined by the remaining keywords
	BATCH	Batch process logout
	DETACHED	Detached process logout
	DIALUP	Dialup interactive process logout
	LOCAL	Local interactive process logout
	NETWORK	Logout by a network server task
	PROCESS	Subprocess or detached process logout
	REMOTE	Logout of a process that logged in interactively from another network node
MOUNT	The issuing of a MOUNT or DISMOUNT request	

/FAILURE_MODE[=keyword]

Specifies how VMS proceeds following a failed attempt to write a security alarm. The following list describes the keywords you can specify with the /FAILURE_MODE qualifier:

Option	Description
WAIT	Indicates that processes are placed in the MWAIT state to wait until the resource is available. This is the default.
IGNORE	Indicates that failing security alarms are to be ignored. The first failed alarm causes an error message to be written to the operator console and log file. The system maintains a count of the lost alarms, which can be displayed with SHOW AUDIT.
CRASH	Forces a system failure if security alarms cannot be written.

example

```
$ SET AUDIT/ALARM/ENABLE=(AUTHORIZATION,BREAKIN)
```

The SET AUDIT command in this example enables alarms at all terminals established as security operators for any change in the system user or network proxy authorization file and for any break-in attempts.

SET BROADCAST

Enables you to selectively screen out various kinds of messages from being broadcast to your terminal.

format

SET BROADCAST=*(class-name[,...])*

parameter

class-name

Specifies the class of message that you want to enable or disable for broadcast to your terminal. If you specify only one class, you can omit the parentheses. The class names are as follows:

ALL	All message classes enabled
[NO]DCL	CTRL/T and SPAWN/NOTIFY messages
[NO]GENERAL	All normal REPLY messages or messages from $BRDCST
[NO]MAIL	Notification of mail
NONE	All message classes disabled
[NO]OPCOM	Messages issued by OPCOM
[NO]PHONE	Messages from the Phone Utility
[NO]QUEUE	Messages referring to print or batch jobs issued by the queue manager
[NO]SHUTDOWN	Messages issued from REPLY/SHUTDOWN
[NO]URGENT	Messages issued from REPLY/URGENT
[NO]USER1 - [NO]USER16	Messages from the specified user groups

example

```
$ SET BROADCAST=NONE
.
.
.
$ SET BROADCAST=(SHUTDOWN, URGENT, DCL, OPCOM)
```

In this example, the first SET BROADCAST command screens out all messages. Later the second SET BROADCAST command restores shutdown, urgent, DCL, and OPCOM messages. General, phone, mail, queue, and user messages are still screened.

SET CARD_READER

Defines the default translation mode for cards read from a card reader. All subsequent input read from the specified card reader are converted using the specified mode.

format

SET CARD_READER *device-name[:]*

parameter

device-name[:]
Specifies the name of the card reader for which the translation mode is to be set. The device must not be currently allocated to any other user.

qualifiers

/026
Sets the card reader for cards punched on an 026 punch.

/029
Sets the card reader for cards punched on an 029 punch.

/LOG
/NOLOG (default)
Controls whether log information is displayed at the terminal to confirm that the card reader is set.

example

```
$ ALLOCATE CR:
  _CRA0: ALLOCATED
$ SET CARD_READER CRA0:/029
$ COPY  CRA0: [MALCOLM.DATAFILES]CARDS.DAT
```

The ALLOCATE command requests the allocation of a card reader by specifying the generic device name. When the ALLOCATE command displays the name of the device, the SET CARD_READER command sets the translation mode at 029. Then the COPY command copies all the cards read by the card reader CRA0 into the file CARDS.DAT in the directory [MALCOLM.DATAFILES].

SET CLUSTER/EXPECTED_VOTES

Sets the total expected votes in the cluster to a value that you specify or, if no value is specified, sets the total votes to a value determined by the system.

Requires OPER privilege and the /EXPECTED_VOTES qualifier.

format

SET CLUSTER/EXPECTED_VOTES *[=value]*

example

```
$ SET CLUSTER/EXPECTED_VOTES=9
```

The SET CLUSTER command in this example sets the total expected votes to 9, which is the value specified in the command string.

SET CLUSTER/QUORUM

Sets the quorum in the cluster to a value that you specify or, if no value is specified, sets the cluster quorum to a value determined by the system. The /QUORUM qualifier is required.

As of VMS Version 5.0, the SET CLUSTER/QUORUM command is superseded by the SET CLUSTER/EXPECTED_VOTES command. DIGITAL recommends the use of the SET CLUSTER/EXPECTED_VOTES command. See SET CLUSTER/EXPECTED_VOTES for a complete description of this command.

Requires OPER privilege.

format

SET CLUSTER/QUORUM *[=quorum-value]*

SET COMMAND

Invokes the Command Definition Utility to add commands to your process command table or to a specified command table file.

format

SET COMMAND *[file-spec[,...]]*

SET CONTROL

Enables or disables CTRL/Y or CTRL/T. CTRL/Y interrupts a command and returns you to the DCL command level. CTRL/T momentarily interrupts a command to print a line of statistics.

SET CONTROL=T requires that SET TERMINAL/BROADCAST be set for the information to be displayed at your terminal.

format

SET [NO]CONTROL*[=(T,Y)]*

parameter

(T,Y)
Specifies that T (CTRL/T) or Y (CTRL/Y) be enabled or disabled. If you specify both characters, separate them with a comma and enclose the list in parentheses. If you do not specify either T or Y, Y is the default.

example

```
$ CTRL/T
NODE22::SMITH  16:21:04  (DCL)   CPU=00:03:29.39 PF=14802 IO=18652 MEM=68
$ SET NOCONTROL=T
$ CTRL/T
```

As shown in this example, when you press CTRL/T, the system displays the appropriate information. The SET NOCONTROL=T command disables the CTRL/T function. Now when you press CTRL/T, no information is displayed.

SET DAY

Sets the default day type specified in the user authorization file (UAF) for the current day.

Requires OPER privilege.

format

SET DAY

qualifiers

/DEFAULT
Overrides any previous SET DAY specification and specifies that the normal UAF defaults are to be used to determine today's day type.

/LOG
/NOLOG (default)
Controls whether log information is displayed at the terminal to confirm that the new SET DAY information has been set.

/PRIMARY
Sets today until midnight to a primary day.

/SECONDARY
Sets today until midnight to a secondary day.

example

```
$ SET DAY/PRIMARY
```

The SET DAY command in this example overrides the current default day type and sets the today until midnight to a primary day.

SET DEFAULT

Sets your default device and directory specifications. The new default is applied to all subsequent file specifications that do not explicitly include a device or directory name.

format

SET DEFAULT *[device-name[:]][directory-spec]*

parameters

device-name[:]
The name of the device you want to go to.

directory-spec
The name of the directory you want to go to. A directory name must be enclosed in brackets. Use the minus sign to specify the next higher directory from the current default.

You must specify either the device-name parameter or the directory-spec parameter. If you specify only the device name, the current directory is the default for the directory-spec parameter. If you specify only the directory name, the current device is the default for the device-name parameter.

example

```
$ SET DEFAULT $FLOPPY1:[WATER.MEMOS]
```

The SET DEFAULT command in this example sets your default to the WATER.MEMOS subdirectory on $FLOPPY1.

SET DEVICE

Establishes a print device or terminal as a spooled device or establishes the status of error-logging for a device.

Requires OPER privilege.

format

SET DEVICE *device-name[:]*

parameter

device-name[:]
Specifies the name of the device whose spooling or error-logging status is to change. The device must be a print device or a terminal if its spooling status is to change; the device must be a disk or magnetic tape if its error-logging status is to change.

qualifiers

/AVAILABLE
/NOAVAILABLE
Controls whether the specified disk is to be considered available. This command can be entered only after the specified disk has been dismounted. If you specify /NOAVAILABLE, any attempt to mount the specified disk is prevented.

/DUAL_PORT
/NODUAL_PORT
Controls whether the port seize logic in the device driver of the specified disk is to be enabled. This qualifier should be used only on disks that contain a dual port kit and have been dismounted.

/ERROR_LOGGING
/NOERROR_LOGGING
Controls whether device errors are logged in the error log file. Use the SHOW DEVICE/FULL command to find out the current status.

/LOG
/NOLOG (default)
Controls whether log information is displayed at the terminal.

/SPOOLED[=(queue-name[:],intermediate-disk-name[:])]
/NOSPOOLED

Controls whether files are spooled to an intermediate disk. The queue name indicates the printer queue to which a file is queued. If a queue name is not supplied, the default is the name of either the printer or terminal. The intermediate disk name identifies the disk to which the spooled files are written. If the intermediate disk name is not supplied, the default is SYS$DISK (the current default disk). The intermediate disk must be mounted before files can be written to it.

example

```
$ SET DEVICE/SPOOLED=(LPA0)  LPA0:
```

In this example, the /SPOOLED qualifier requests that the printer queue LPA0 be spooled to an intermediate disk before files directed to the disk are printed. Because no intermediate disk was specified, the intermediate disk defaults to SYS$DISK.

SET DEVICE/ACL

Allows you to modify the access control list (ACL) of a device. The /ACL qualifier is required.

As of VMS Version 5.0, the SET DEVICE/ACL command is superseded by the SET ACL command. SET DEVICE/ACL is synonymous with SET ACL/OBJECT_TYPE=DEVICE. DIGITAL recommends usage of the SET ACL command.

format

SET DEVICE/ACL*[=(ace[,...])] device-name*

SET DEVICE/SERVED

Allows you to make a disk on a local node available to all the nodes in a cluster. The /SERVED qualifier is required.

Applies only to VAXcluster environments.

format

SET DEVICE/SERVED *node-name$DDcu:*

parameter

node-name$DDcu:
Specifies the device name of the device that you want to make available to the cluster.

description

The SET DEVICE/SERVED command is used in conjunction with the Mass Storage Control Protocol (MSCP) server to make a disk on a local node available to all nodes on the cluster. The local node must be a member of a VAXcluster, and the local MSCP server must have been invoked by the SYSGEN Utility.

NOTE: Unless the disk device that you intend to make available to the cluster is a system disk, it must not already be mounted when you enter the SET DEVICE/SERVED command.

The SET DEVICE/SERVED command string can be included as part of the local startup command file, and entered before the Mount Utility mounts the disk to be served (made available to the entire cluster).

example

```
$ SET DEVICE/SERVED DRA4:
```

The SET DEVICE/SERVED command in this example instructs the MSCP server to make the disk device DRA4 on your local node available to all other processors on your cluster.

SET DIRECTORY

Modifies the characteristics of one or more directories.

See the qualifier descriptions for restrictions.

format

SET DIRECTORY *[device-name[:]]directory-spec[,...]*

parameters

device-name[:]
Specifies the device on which the directory that you want to modify is located. The device name parameter is optional.

directory-spec[,...]
Specifies one or more directories to be modified. If you specify two or more directories, separate them with commas. Wildcard characters are allowed.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each SET DIRECTORY operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and RETURN. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the SET DIRECTORY operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/LOG
/NOLOG (default)

Controls whether the system displays the directory specification of each directory that is modified as the command executes.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/OWNER_UIC[=uic]

Requires SYSPRV privilege to specifiy a UIC other than your own.
Specifies an owner UIC for the directory. The default UIC is that of the current process.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/VERSION_LIMIT[=n]

Specifies the total number of versions that a file in the specified directory can have. If you do not specify a version limit, a value of 0 is used, indicating that the number of versions of a file is limited only to the Files-11 architectural limit—32,767. If you change the version limit for the directory, the new value applies only to files created after the change has been made.

example

```
$ SET DIRECTORY/VERSION_LIMIT=5/CONFIRM [SMITH...]
```

The SET DIRECTORY command in this example sets a version limit of five for all files in the SMITH directory and all subdirectories of [SMITH]. The /CONFIRM qualifier requests that you confirm whether or not the specified directory should actually be modified. Note that it only affects the files created after the command is entered.

SET DIRECTORY/ACL

Allows you to modify the access control list (ACL) of one or more directories. The /ACL qualifier is required.

As of VMS Version 5.0, the SET DIRECTORY/ACL command is superseded by the SET ACL command. SET DIRECTORY/ACL is synonymous with SET ACL/OBJECT_TYPE=DIRECTORY. DIGITAL recommends usage of the SET ACL command.

format

SET DIRECTORY/ACL*[=(ace[,...])] directory-spec[,...]*

SET ENTRY

Changes the current status or attributes of a job that is not currently executing in a queue.

Requires OPER privilege or EXECUTE (E) access to the specified queue. If you have DELETE (D) access to the specified job, you can alter the attributes for that job.

format

SET ENTRY *entry-number*

parameter

entry-number
Specifies the entry number of the job you want to change. The job number is displayed at the time of the job's submission.

qualifiers

/AFTER=time
/NOAFTER
Requests that the specified job be held until after a specific time. If the specified time has already passed, the job is queued for immediate

processing. You can specify either an absolute time or a combination of absolute and delta times.

/BURST[=keyword]
/NOBURST

Controls whether a burst page is included at the beginning of a print job. A burst page precedes a flag page and contains the same information. However, it is printed over the perforation between the burst page and the flag page. The printing on the perforation makes it easy to separate individual print jobs. When you specify /BURST, you need not specify /FLAG; a flag page automatically follows the burst page. Possible keywords are as follows:

ALL	All printed files contain a burst page.
ONE	The first printed file contains a burst page.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS

Enables you to change the characteristics desired for the job. If you specify only one characteristic, you can omit the parentheses. Codes for characteristics can be either names or values from 0 to 127 and are installation-defined. Use the SHOW QUEUE/CHARACTERISTICS command to see which characteristics have been defined for your system. Use the SHOW QUEUE/FULL command to see which characteristics are available on a particular queue.

/CLI=filename

Specifies the name of a command language interpreter (CLI) to use in processing the job. The file name specifies that the CLI be SYS$SYSTEM:filename.EXE. If you do not specify the /CLI qualifier, the job is run by the CLI specified in the user's authorization record, or whatever CLI was specified when the job was originally submitted to the queue.

/COPIES=n

Specifies the number of copies to print. The n parameter can be any number from 1 to 255. When you use the /COPIES qualifier with the SET ENTRY command, the number of copies can apply only to the entire job. You cannot use this qualifier to specify different numbers of copies for individual files within a multifile job.

/CPUTIME=option

Specifies a CPU time limit for the batch job. Time can be specified as: delta time, 0, NONE, or INFINITE. Both the value 0 and the keyword INFINITE allow unlimited CPU time; the keyword NONE defaults to your UAF value or the limit specified on the queue. You cannot specify more time than permitted by the base queue limits or your own UAF.

/FEED
/NOFEED

Controls whether form feeds are inserted into print jobs when the printer nears the end of a page. The number of lines per form can be reset by the /FORM qualifier. You can suppress this automatic form feed (without affecting any of the other carriage control functions that are in place) by using the /NOFEED qualifier. When you use the /FEED qualifier with the SET ENTRY command, the qualifier applies to all files in the print job. You cannot use this qualifier to specify form feeds for individual files within a multifile job.

/FLAG[=keyword]
/NOFLAG

Controls whether a flag page is printed preceding a print job. The flag page contains the name of the user submitting the job, the job entry number, and other information about the job. You can specify one of the following keywords:

ALL	Prints a flag page before each file in the job
ONE	Prints a flag page before the first file in the job

/FORM=type

Specifies the name of the form that you want for the print job. Specify the form type using a numeric value or alphanumeric code. Form types can refer to the width, length, or type of paper. Codes for form types are installation-defined. You can use the SHOW QUEUE/FORM command to find out the form types available for your system. The SHOW QUEUE/FULL command tells you which form is set for a specific queue. If you specify a form type different from that of the queue, your job remains pending until the form type of the queue is set equal to the form type of the job or you delete the job with the DELETE/ENTRY command. You can use the SET ENTRY command to change the form type of your job to match that of the queue. To change the form type for the queue, stop the queue, physically change the form, and restart the queue, specifying the new form type.

/HEADER
/NOHEADER

Controls whether a heading line is printed at the top of each output page in a print job.

/HOLD
/NOHOLD

Controls whether or not the job is to be made available for immediate processing or held for processing later. If you specify /HOLD, the job is not released for processing until you specifically release it with the /NOHOLD or /RELEASE qualifier. You can use the SET ENTRY command to release a job that was previously submitted with a /HOLD qualifier, or you can place a job on hold so that it will run later.

/JOB_COUNT=n

Requests that an entire print job be printed n times, where n is a decimal integer from 1 to 255. This qualifier overrides the /JOB_COUNT qualifier specified or defaulted with the PRINT command.

/KEEP
/NOKEEP

Controls whether the batch job log file is deleted after it is printed.

/LOG_FILE=file-spec
/NOLOG_FILE

Creates a log file with the specified file specification. You can specify a different device name, as long as the process executing the batch job has access to the device on which the log file will reside. Logical names in the file specification are translated in the context of the process that executes the SET ENTRY command. If you omit the /LOG_FILE qualifier and specify the /NAME qualifier, the log file is written to a file having the same file name as that specified by the /NAME qualifier; the file type is LOG. When you omit the /LOG_FILE qualifier, the job-name value used with /NAME must be a valid file name.

/LOWERCASE
/NOLOWERCASE

Indicates whether the files must be printed on a printer that can print both uppercase and lowercase letters. The /NOLOWERCASE qualifier means that files can be printed on printers supporting only uppercase letters. If all available printers can print both uppercase and lowercase letters, you do not need to specify /LOWERCASE.

/NAME=job-name

Defines a name string to identify the job. The name string can have from 1 to 39 characters. The job name is used in the SHOW QUEUE command display. For batch jobs, the job name is also used for the batch job log file. For print jobs, the job name is also used on the flag page of the printed output. If the /NAME qualifier has not been specified for the job, the name string defaults to the file name of the first, or only, file in the job; the file type is LOG.

/NOCHECKPOINT

For a batch job, erases the value established by the most recently executed SET RESTART_VALUE command. For a print job, clears the stored checkpoint so that the job will restart from the beginning.

/NODELETE

Cancels file deletion for a job that was submitted with the /DELETE qualifier. If no /DELETE qualifier was specified when the job was originally submitted to the queue, you cannot use the SET ENTRY to establish file deletion at a later time. You cannot use the /NODELETE qualifier to specify that individual files in a multifile job not be deleted.

/NOTE=string

Specifies a message of up to 255 characters to appear on the flag page of the job. Enclose the message in quotation marks if it contains spaces, special characters, or lowercase characters.

/NOTIFY
/NONOTIFY

Controls whether a message is broadcast to any terminal at which you are logged in, notifying you when your job has been completed or aborted.

/OPERATOR=string

Specifies a message string of up to 255 characters to be sent to the operator just before the job begins execution. When the job begins execution, the queue pauses and the message is transmitted to the operator. Enclose the message in quotation marks if it contains spaces, special characters, or lowercase characters.

/PAGES=([l,]u)

Specifies the number of pages to print for the specified job. You can use the /PAGES qualifier to print portions of a long file. When you use the /PAGES qualifier with the SET ENTRY command, the qualifier can only apply to an entire job. You cannot use this qualifier to specify different numbers of pages to be printed for individual files within a multifile job. The l (lower) specifier refers to the first page in the group of pages that you want printed for that job. If you omit the l specifier, the printing starts on the first page of the job. The u (upper) specifier refers to the last page of the file that you want printed. You must use two consecutive quotation marks ("") if you omit the upper parameter.

/PARAMETERS=(parameter[,...])

Specifies from 1 to 8 optional parameters to be passed to the job. Each parameter can have as many as 255 characters. If you specify only one parameter, you can omit the parentheses. The commas delimit individual parameters. To specify a parameter that contains any special characters or delimiters, enclose the parameter in quotation marks. For batch jobs, the parameters define values to be equated to the symbols named P1 through P8 in each command procedure in the job. The symbols are local to the specified command procedures.

/PASSALL
/NOPASSALL

Specifies whether the symbiont bypasses all formatting and sends the output QIO to the driver with format suppressed. All qualifiers affecting formatting, as well as the /HEADER, /PAGES, and /PAGE_SETUP qualifiers, will be ignored. When you use the /PASSALL qualifier with the SET ENTRY command, the qualifier applies to the entire job. You cannot use this qualifier to specify PASSALL mode for individual files within a multifile job.

/PRINTER[=queue-name]
/NOPRINTER

Queues the batch job log to the specified printer queue when the job is completed. By default, the printer queue for the log file is SYS$PRINT. The /PRINTER qualifier allows you to specify a particular printer queue. The /NOPRINTER qualifier assumes the /KEEP qualifier.

/PRIORITY=n

Requires OPER or ALTPRI privilege to raise the priority above the value of the SYSGEN parameter MAXQUEPRI. Specifies the priority of the job. The priority value must be in the range of 0 through 255, where 0 is the lowest priority and 255 is the highest. The default value for /PRIORITY is the value of the SYSGEN parameter DEFQUEPRI. No privilege is needed to set the priority lower than the MAXQUEPRI value.

/RELEASE

Releases for processing jobs submitted with the /HOLD qualifier or /AFTER qualifier, jobs held in a queue with the /RETAIN qualifier, and jobs refused by a user-written symbiont.

/REQUEUE=queue-name[:]

Requests that the job be moved from the original queue to the specified queue.

/RESTART
/NORESTART

Specifies whether a batch or print job will be restarted after a system crash or a STOP/QUEUE/REQUEUE command.

/SETUP=module[,...]

Extracts the specified module from the device control library (containing escape sequence modules for programmable printers) and copies the module to the printer before a file is printed. When you use the /SETUP qualifier with the SET ENTRY command, the qualifier applies to the entire job. You cannot use this qualifier to specify different setup modules for individual files within a multifile job.

/SPACE
/NOSPACE

The /SPACE qualifier specifies that the output is to be double-spaced. Specifying /NOSPACE causes the output to be single-spaced. When you use the /SPACE qualifier with the SET ENTRY command, the qualifier applies to the entire job. You cannot use this qualifier to specify different spacing for individual files within a multifile job.

/TRAILER[=keyword]
/NOTRAILER

Controls whether a trailer page is printed at the end of a job. The trailer page displays the job entry number, as well as information about the user submitting the job. When you use the /TRAILER qualifier with the SET

ENTRY command, trailer pages are placed at the end of each file in a multifile job. Possible keywords are as follows:

ALL	All printed files contain a trailer page
ONE	The last printed file contains a trailer page

/WSDEFAULT=n

Defines a working set default for a batch job. Possible values are a positive integer in the range 1 through 65,535, 0, or the keyword NONE (the default) for n. Use this qualifier to override the base queue value established by the system manager or the value authorized in the user authorization file (UAF), provided you want to impose a lower value. Specify 0 or NONE if you want the working set value defaulted to either the UAF value or the working set quota specified on the queue. You cannot request a value higher than the default.

/WSEXTENT=n

Defines a working set extent for a batch job. Possible values are a positive integer in the range 1 through 65,535, 0, or the keyword NONE (the default) for n. Use this qualifier to override the base queue value established by the system manager or the value authorized in the user authorization file (UAF), provided you want to impose a lower value. Specify 0 or NONE if you want the working set extent defaulted to either the UAF or the working set extent specified on the queue. You cannot request a value higher than the default.

/WSQUOTA=n

Defines the maximum working set size for a batch job. This is the working set quota. Possible values are: a positive integer in the range 1 through 65,535, 0, or the keyword NONE (the default) for n. Use this qualifier to override the base queue value established by the system manager or the value authorized in the user authorization file (UAF), provided you want to impose a lower value. Specify 0 or NONE if you want the working set quota defaulted to either the user authorization file value or the working set quota specified on the queue. You cannot request a value higher than the default.

example

```
$ PRINT/HOLD   MYFILE.DAT
  Job MYFILE (queue SYS$PRINT, entry 112) holding
    .
    .
    .
$ SET ENTRY 112/RELEASE/JOB_COUNT=3
```

The PRINT command in this example requests that the file MYFILE.DAT be queued to the system printer, but placed in a hold status. The SET ENTRY command releases the file for printing and changes the number of copies of the job to three.

SET FILE

Modifies the characteristics of one or more files.

See the qualifier descriptions for restrictions.

format

SET FILE *file-spec[,...]*

parameter

file-spec[,...]
Specifies one or more files to be modified. If you specify two or more files, separate them with commas. Wildcard characters are allowed.

qualifiers

/ACL
Modifies an access control list (ACL) associated with one or more files. For more information, see the description of the SET FILE/ACL command.

/BACKUP
/NOBACKUP
Specifies that BACKUP records the contents of the file. The /NOBACKUP qualifier causes BACKUP to record the attributes of the file but not its contents. Valid only for Files–11 Structure Level 2 files.

/BEFORE[=time]
Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/BY_OWNER[=uic]
Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)
Controls whether a request is issued before each SET FILE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for

example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED
Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation.

/DATA_CHECK[=([NO]READ,[NO]WRITE)]
Specifies whether a READ data check (rereading each record), a WRITE data check (reading each record after it is written), or a combination of the two is performed on the file during transfers. By default, a WRITE data check is performed.

/END_OF_FILE
Resets the end-of-file mark to the highest block allocated.

/ENTER=new-file-spec
Use with caution. Assigns an additional name to a single file so that the file has a second name, or alias. However, both the original name and the alias reference the same file. For this reason, take care when deleting files that have aliases. To keep the file but remove one of its names, use the /REMOVE qualifier with SET FILE. No wildcards are allowed in the file specification.

/ERASE_ON_DELETE
Specifies that the specified files are erased from the disk (not just merely written over) when the DELETE or PURGE command is issued for the files. See DELETE/ERASE for more information.

/EXCLUDE=(file-spec[,...])
Excludes the specified file from the SET FILE operation. You can include a directory name but not a device name in the file specifications. Wildcard characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRATION_DATE=date
/NOEXPIRATION_DATE
Requires ownership of the file or access control. Controls whether an expiration date is assigned to the specified files. Absolute date keywords are allowed. If you specify 0 as the date, today's date is used.

/EXTENSION[=n]

Sets the extend quantity default for the file. The value of *n* can range from 0 through 65,535. If you omit the value specification or specify a value of 0, VMS RMS calculates its own /EXTENSION value.

/GLOBAL_BUFFER=n

Sets the VAX RMS global buffer count (the number of buffers that can be shared by processes accessing the file) for the specified files. The value *n* must be an integer in the range 0 through 32,767. A value of 0 disables buffer sharing.

/LOG
/NOLOG (default)

Displays the file specification of each file modified as the command executes.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with /CREATED, which also allows you to select files according to time attributes. If you do not specify /MODIFIED, the default is /CREATED.

/NODIRECTORY

Use with extreme caution. This qualifier removes the directory attributes of a file and allows you to delete the corrupted directory file even if other files are contained in the directory. When you delete a corrupted directory file, the files contained within it are lost. Use ANALYZE/DISK_STRUCTURE /REPAIR to place the lost files in [SYSLOST]. You can then copy the lost files to a new directory. This qualifier is valid only for the Files–11 Structure Level 2 files.

/OWNER_UIC[=uic]

Requires GRPPRV to set the owner to another member of the same group. Requires SYSPRV to set the owner to any UIC outside your group. Specifies an owner user identification code (UIC) for the file. The default is the UIC of your process.

/PROTECTION[=(code)]

Cannot be used to change the protection on a file via DECnet. Enables you to change or reset the protection for one or more of your files. The ownership categories are SYSTEM, OWNER, GROUP, AND WORLD. The access categories are R (read), W (write), E (execute), and D (delete). If you specify /PROTECTION without the ownership and access code, the file protection is set according to the current default protection.

/REMOVE

Use with caution. This qualifier enables you to remove one of the names of a file that has more than one name, without deleting the file. If you have created an additional name for a file with the /ENTER qualifier of SET FILE,

you can use the /REMOVE qualifier to remove either the original name or the alias.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/STATISTICS
/NOSTATISTICS (default)
Enables the gathering of RMS statistics on the specified file. These statistics can subsequently be viewed using the Monitor Utility, which is invoked with the DCL command MONITOR.

/TRUNCATE
Truncates the file at the end of the block containing the end-of-file (EOF) marker, that is, releases allocated but unused blocks of the file.

/UNLOCK
Makes one or more improperly closed files accessible.

/VERSION_LIMIT[=n]
Specifies the maximum number of versions for the specified file. If you do not specify a version limit, a value of 0 is used, indicating that the number of versions of a file is limited only to the Files–11 architectural limit of 32,767. When you exceed that limit, the earliest version of the file is deleted from the directory without notification to the user.

example

```
$ SET FILE/EXPIRATION_DATE=31-DEC-1988:11:00 BATCH.COM;3
```

The SET FILE command requests that the expiration date of the file BATCH.COM;3 be set to 11:00 a.m., December 31, 1988.

SET FILE/ACL

Allows you to modify the access control list (ACL) of one or more files. The /ACL qualifier is required.

As of VMS Version 5.0, the SET FILE/ACL command is superseded by the SET ACL command. SET FILE/ACL is synonymous with SET ACL /OBJECT_TYPE=FILE. DIGITAL recommends usage of the SET ACL command.

format

SET FILE/ACL*[=(ace[,...])] file-spec[,...]*

SET HOST

Connects your terminal (through the current host processor) to another processor, called the remote processor. Both processors must be running DECnet. [1]

- **You can use the SET HOST command only if your system is connected by DECnet to another system.**
- **You must have an account on the remote system to log in after the SET HOST command has made the connection.**
- **The SET HOST command requires the network mailbox privilege NETMBX.**

format

SET HOST *node-name*

parameter

node-name
Specifies the node name of the remote processor to which you will connect.

qualifiers

/BUFFER_SIZE=n
Changes the packet size of the protocol message sent between the terminal and the remote processor if a connection to the remote processor is already established. The default buffer size is 1010 bytes; however, *n* can range from 140 bytes to 1024 bytes. The value of *n* is reset to 140 bytes if a value below 140 is specified; a value for *n* above 1024 bytes is reset to 1024.

/LOG[=file-spec]
/NOLOG (default)
Controls whether a log file of the entire session is kept. If you use /LOG without the file specification, the log information is stored in the file SETHOST.LOG.

/RESTORE
/NORESTORE
Saves current terminal characteristics before a remote terminal session is begun and restores them when the remote session is terminated.

---

[1] Available under separate license.

example

```
$ SET HOST ITALIC
Username:  BROWN
Password:
    Welcome to VAX/VMS Version 5.0 on node ITALIC
    .
    .
    .
$  LOGOUT
BROWN logged out at 31-DEC-1988 15:04:25.27
%REM-S-END, Control returned to node _CASLON::
```

In this example, the name of the local node is CASLON. This SET HOST command connects the user terminal to the processor at the network node named ITALIC. The remote processor then prompts for user name and password. Use the normal login procedure to log in to the remote processor.

Once you are logged in at a remote node, you can use the SET HOST command to establish communication with another node. After logging into node ITALIC, you could type SET HOST BODONI. You would again be prompted for a user name and password. If you then supply a valid user name and password, you will be logged in at node BODONI. Note that when you log out at node BODONI, control is returned to node ITALIC. You must log out from node ITALIC to return to your local node, CASLON.

SET HOST/DTE

Connects your system to a remote system through an out-going terminal line. Exit from the remote system by typing CTRL/\; that is, type a backslash (\) while holding down the CTRL key.

You must have an account on the remote system in order to log in to that system after the connection is made. Also requires the ability to assign a channel to the terminal port specified. By default, BYPASS privilege is required but can be changed by setting the device protection for the terminal port.

format

SET HOST/DTE *terminal-name*

parameter

terminal-name
Specifies the name of an out-going terminal line, which connects your system directly to another system or to a modem.

qualifiers

/DIAL=(NUMBER:number[,MODEM_TYPE:modem-type])
Allows a modem attached to the out-going terminal line to be autodialed using the autodial protocol of that modem. The NUMBER keyword is the telephone number to be autodialed and is a required parameter. The MODEM_TYPE: keyword is optional and can be used to specify a modem-type of DF03, DF112 or DMCL. By default, a modem-type of DF03 is assumed. DMCL is any modem that uses the DEC Modem Command Language.

/LOG[=file-spec]
/NOLOG
Controls whether a log file of the entire session is kept. If you do not specify a file, the log information is stored in the file SETHOST.LOG.

example

```
$ SET HOST/DTE/DIAL=(NUMBER:5551234#,MODEM_TYPE:DF112) TTA2:
Username: SMITH
Password:
```

The SET HOST/DTE command in this example accomplishes the same thing as in the first example, except that it uses the DF112 modem. Note that the number sign (#) is required to activate the autodialer in the DF112.

SET HOST/DUP

Connects your terminal to a storage controller through the appropriate bus for that controller.

For use only with storage controllers. Requires the DIAGNOSE privilege.

format

SET HOST/DUP*/SERVER=server-name*
/TASK=task-name node-name

parameter

node-name
Specifies the node name of the storage controller.

qualifiers

/LOG[=file-spec]
/NOLOG (default)
Controls whether a log file of the entire session is kept. If you use /LOG without the file specification, the log information is stored in the file HSCPAD.LOG.

/SERVER=server-name
Specifies the server name for the target storage controller.

This qualifier is required.

/TASK=task-name
Specifies the utility or diagnostic name to be executed on the target storage controller under direction of the server.

This qualifier is required.

example

```
$ SET HOST/DUP/SERVER=DUP$/TASK=DIRECT BLKHOL
%HSCPAD-I-LOCPROGEXE, Local program executing - type ^\ to exit utility
```

The SET HOST/DUP command in this example connects the user terminal to the utility program called DIRECT executing on a storage controller named BLKHOL under direction of the DUP$ server.

SET HOST/HSC

Connects your terminal to a remote HSC50 disk and tape controller through the Computer Interconnect bus.

Used only with remote HSC50s. Requires the DIAGNOSE privilege.

format

SET HOST/HSC *node-name*

parameter

node-name
Specifies the node name of the remote HSC50.

qualifier

/LOG[=file-spec]
/NOLOG (default)
Controls whether a log file of the entire session is kept. If you use /LOG without the file specification, the log information is stored in the file HSCPAD.LOG.

example

```
$ SET HOST/HSC HSC001
%HSCPAD-I-LOCPROGEXE, Local program executing - type ^\ to exit, ^Y for prompt
HSC50>
```

This SET HOST/HSC command connects the user terminal to the HSC named HSC001.

SET KEY

Sets and locks the key definition state for keys defined with the DEFINE /KEY command.

format

SET KEY

qualifiers

/LOG (default)
/NOLOG
Controls whether the system displays a message indicating that the key state has been set.

/STATE=state-name
/NOSTATE
Specifies the name of the state. The state name can be any alphanumeric string. If you omit the /STATE qualifier or use /NOSTATE, the current state is left unchanged. The default state is DEFAULT.

example

```
$ SET KEY /STATE=EDITING
```

The SET KEY command in this example sets the key state to the EDITING state. You can now use the key definitions that were defined for the EDITING state.

SET LOGINS

Sets the interactive limit (number of interactive users allowed on the system), or displays the interactive limit and the current number of interactive users.

Requires OPER privilege.

format

SET LOGINS

parameters

None.

qualifier

/INTERACTIVE[=n]
Establishes the number of interactive users allowed to gain access to the system. If n is specified, the interactive limit is set to n. If n is not specified, the SET LOGINS command displays the current interactive limit and the number of interactive users.

example

```
$ SET LOGINS/INTERACTIVE=5
%SET-T-INTSET, login interactive limit=5, current interactive value=3
```

In this example, the SET LOGINS command specifies that only five interactive users can be logged in to the system.

SET MAGTAPE

Defines the default characteristics associated with a specific magnetic tape device for subsequent file operations.

The SET MAGTAPE command is valid for magnetic tape devices mounted with foreign volumes.

format

SET MAGTAPE *device-name[:]*

parameter

device-name[:]
Specifies the name of the magnetic tape device for which the characteristics are to be set. The device must not be currently allocated to any other user.

qualifiers

/DENSITY=density
Specifies the default density, in bits per inch (bpi), for all write operations on the magnetic tape device when the volume is mounted as a foreign tape or as an unlabeled tape. The density can be specified as 800, 1600, or 6250, if supported by the magnetic tape drive.

/END_OF_FILE
Writes a tape mark at the current position on the magnetic tape volume.

/LOG
/NOLOG
Displays information about the operations performed on the magnetic tape volume.

/LOGSOFT (default)
/NOLOGSOFT
Controls whether soft errors on the specified device are to be logged in the error log file. Soft errors are errors corrected by the hardware without software intervention. This qualifier only affects devices that support hardware error correction, such as the TU78 magnetic tape drive. When used with other devices, this qualifier has no effect.

/REWIND

Requests that the volume on the specified device be rewound to the beginning of the magnetic tape.

/SKIP=option

Requests that the magnetic tape volume be positioned according to any of the following options:

BLOCK:n	Directs the SET MAGTAPE command to skip the specified number of blocks
END_OF_TAPE	Directs the SET MAGTAPE command to position the volume at the end-of-tape mark
FILES:n	Directs the SET MAGTAPE command to skip the specified number of files
RECORD:n	Directs the SET MAGTAPE command to skip the specified number of records

/UNLOAD

Requests that the volume on the specified device be rewound and unloaded.

example

```
$ MOUNT MTB1:/FOREIGN
$ SET MAGTAPE MTB1: /DENSITY=800
```

The MOUNT command in this example mounts a foreign tape on the device MTB1. The SET MAGTAPE command defines the density for writing the magnetic tape at 800 bpi.

SET MESSAGE

Sets the format for system messages or specifies a process level message file. Lets you override or supplement the system messages.

format

SET MESSAGE *[file-spec]*

parameter

file-spec

Specifies the name of the process level message file. Messages in this file supersede messages for the same conditions in the system message file or in an existing process message file. The file type defaults to EXE. No wildcard characters are allowed. If you do not specify this parameter, the qualifiers apply to the system message file.

qualifiers

/DELETE
Removes any process permanent message files currently in effect. Do not specify the file-spec parameter with the /DELETE qualifier.

/FACILITY (default)
/NOFACILITY
Formats messages so that the facility name prefix appears.

/IDENTIFICATION (default)
/NOIDENTIFICATION
Formats messages so that the message identification prefix appears.

/SEVERITY (default)
/NOSEVERITY
Formats messages so that the severity level appears.

/TEXT (default)
/NOTEXT
Formats messages so that the message text appears.

example

```
$ SET MESSAGE/TEXT/NOFACILITY/NOIDENTIFICATION/NOSEVERITY
$ SHOW DEVICES/MUONTED
unrecognized qualifier - check validity, spelling, and placement
\MUONTED\
```

The SET MESSAGE command in this example formats the error message so that only the text appears.

SET ON

Enables error checking by the command interpreter after the execution of each command in a command procedure. Specify SET NOON to disable error checking.

format

SET [NO]ON

parameters

None.

description

Use the SET NOON command to override default error checking. When SET NOON is in effect, the command interpreter continues to place the status code value in $STATUS and the severity level in $SEVERITY, but does not perform any action based on the values. As a result, the command procedure continues to execute no matter how many errors are returned. The SET ON or SET NOON command applies only at the current command level.

example

```
$ SET NOON
$ DELETE  *.SAV;*
$ SET ON
$ COPY  *.OBJ  *.SAV
```

This command procedure routinely copies all object modules into new files with the file type SAV. The DELETE command first deletes all existing files with the SAV file type, if any. The SET NOON command ensures that the procedure continues executing even if there are no files with the SAV file type in the current directory. Following the DELETE command, the SET ON command restores error checking. Then the COPY command makes copies of all existing files with OBJ file type.

SET OUTPUT_RATE

Sets the rate at which output is written to a batch job log file.

For use only within command procedures that are submitted as batch jobs.

format

SET OUTPUT_RATE*[=delta-time]*

parameter

delta-time

The time interval at which output is written from the output buffer to the batch job log file. If no delta time is specified, the information is written in the output buffer to the log file, but the output rate is not changed from the default of once per minute. Specify *delta-time* as [dddd-][hh:mm:ss.cc].

example

```
$ SET OUTPUT_RATE=:0:30
```

This command, when executed within a batch job, changes the default output rate from once a minute to once every 30 seconds.

SET PASSWORD

Establishes, changes, or removes a password. SET PASSWORD can be used by users to change their own passwords and by system managers to change the system password.

See the qualifier descriptions for restrictions.

format

SET PASSWORD

parameters

None.

description

All user accounts on a system have passwords. A password is required for logging in to the system. A password contains up to 31 alphanumeric characters. The dollar sign ($) and underscore (_) are also permitted. Uppercase and lowercase characters are equivalent. All lowercase characters are converted to uppercase before the password is encrypted. (For example, *EAGLE* is the same as *eagle*.)

Use the following procedure to change your password:

1. Enter the SET PASSWORD command.
2. The system prompts you for your current password. Enter your current password.
3. The system prompts you for a new password. Enter a new password, or press the RETURN key to disable your current password.
4. The system prompts you to verify the password. Enter the new password to verify. (If the two entries of the new password do not match, the password does not change.)

The following guidelines are recommended to minimize the chances of passwords being discovered by trial-and-error or by exhaustive search:

- Make passwords at least six characters long.
- Avoid names or words that are readily associated with you.
- Change your passwords at least once every month.

To ensure that the above guidelines are met, use the /GENERATE[=value] qualifier. This qualifier generates random passwords of up to 12 characters in length. The system manager can require individual users to use the /GENERATE qualifier.

qualifiers

/GENERATE[=value]

Generates a list of 5 random passwords. Press RETURN to repeat the procedure until a suitable password appears. If no value is specified, SET PASSWORD uses a default value of 6, and generates passwords from 6 to 8 characters long. Values greater than 10 are not accepted and produce errors.

/SECONDARY

Creates or allows you to replace a secondary password. The procedure is the same as setting your primary password. To remove your secondary password, press the RETURN key when SET PASSWORD/SECONDARY prompts you for a new password and verification. Secondary passwords make it possible to set up an account that requires two different people to access it. Each person knows one of the two passwords, and both passwords are required to successfully log in. The /SECONDARY and /SYSTEM qualifiers are incompatible.

/SYSTEM

Requires both SECURITY and CMKRNL privileges. Changes the system password, rather than a user password. The /SYSTEM and /SECONDARY qualifiers are incompatible. Refer to the *Guide to VMS System Security* for more information about the use of system passwords.

example

```
$ SET PASSWORD
Old password: HONCHO
New password: BIG_ENCHILADA
Verification: BIG_ENCHILADA
```

In response to the SET PASSWORD command, the system first prompts for the old password and then for the new password. The system then prompts again for the new password to verify it. The password changes if the user is authorized to change this account's password, if the old password is given correctly, and if the new password is given identically twice. Otherwise, an error message appears and the password remains unchanged.

In a real session, neither the old password nor the new password and its verification appear on the screen or paper.

SET PRINTER

Establishes the characteristics of a specific line printer. The default values listed for qualifiers to the SET PRINTER command are the defaults for an initially bootstrapped system.

Requires OPER privilege. If the printer is a spooled device, the logical I/O privilege (LOG_IO) is required to modify its characteristics.

format

SET PRINTER *printer-name[:]*

parameter

printer-name[:]
Specifies the name of a line printer to set or modify its characteristics. If the printer has been set to /SPOOLED, the logical I/O privilege (LOG_IO) is required to modify its characteristics.

qualifiers

/CR
/NOCR (default)
Controls whether the printer driver outputs a carriage return character. Use this qualifier for printers on which line feeds do not imply carriage returns.

/FALLBACK
/NOFALLBACK (default)
Determines whether or not the printer attempts to translate characters belonging to the DEC Multinational Character Set into 7-bit equivalent representations. If a character cannot be translated, an underscore character is substituted.

/FF (default)
/NOFF
Indicates whether the printer performs a mechanical form feed.

/LA11
Specifies the printer as an LA11.

/LA180
Specifies the printer as an LA180.

/LOG
/NOLOG (default)
Determines whether information confirming the printer setting is displayed at the terminal from which the SET PRINTER command was entered.

/LOWERCASE
/NOLOWERCASE
Indicates whether the printer prints both uppercase and lowercase letters or only uppercase. When the operator specifies the /NOLOWERCASE qualifier, all letters are translated to uppercase.

/LP11 (default)
Specifies the printer as an LP11.

/PAGE=lines-per-page
Establishes the number of lines per page on the currently installed form; the number of lines can range from 1 to 255 and defaults to 64. The printer driver uses this value to determine the number of line feeds that must be entered to simulate a form feed.

/PASSALL
/NOPASSALL (default)
Controls whether the system interprets special characters or passes them as 8-bit binary data.

/PRINTALL
/NOPRINTALL (default)
Controls whether the line printer driver outputs printable 8-bit multinational characters.

/TAB
/NOTAB (default)
Controls how the printer handles TAB characters. The /NOTAB qualifier expands all tab characters to spaces and assumes tab stops at eight character intervals.

/TRUNCATE (default)
/NOTRUNCATE
Controls whether the printer truncates data exceeding the value specified by the /WIDTH qualifier. Note that the /TRUNCATE and /WRAP qualifiers are incompatible.

/UNKNOWN
Specifies the printer as nonstandard.

/UPPERCASE
/NOUPPERCASE
Indicates whether the printer prints both uppercase and lowercase letters or only uppercase ones. When you specify /UPPERCASE, all letters are translated to uppercase.

/WIDTH=n
Establishes the number of characters per output line on currently installed forms. The width, *n*, can range from 0 through 65535 for LP11 controllers, and from 0 through 255 for DMF32 controllers. The default value is 132 characters per line.

/WRAP
/NOWRAP (default)
Controls whether the printer generates a carriage return/line feed when it reaches the end of a line.

example

```
$ SET PRINTER/PAGE=60/WIDTH=80  LPA0:
```

The SET PRINTER command in this example establishes the size of an output page as 60 lines and the width of a line as 80 characters for printer LPA0.

SET PROCESS

Changes the execution characteristics associated with the specified process for the current terminal session or job. If no process is specified, changes are made to the current process.

Requires GROUP privilege to change other processes in the same group. Requires WORLD privilege to change processes outside your group.

format

SET PROCESS *[process-name]*

parameter

process-name
Requires that you own the process or that you have GROUP privilege and that the process is in your group. Specifies the name of the process for which the characteristics are to be changed. The process name can contain from 1 to 15 alphanumeric characters. The default is the current process. Compatible only with the /PRIORITY, /RESUME, and /SUSPEND qualifiers.

qualifiers

/DUMP
/NODUMP (default)
Causes the contents of the address space to be written to the file named SYS$LOGIN:IMAGEDUMP.DMP when an image terminates due to an unhandled error.

/IDENTIFICATION=pid

Requires GROUP or WORLD privilege for processes other than your own. Specifies the process identification value (PID) of the process for which characteristics are to be changed. Overrides the process-name parameter. Compatible only with the /PRIORITY, /RESUME, and /SUSPEND qualifiers.

/NAME=string

Changes the name of the current process to a string of 1 through 15 characters.

/PRIORITY=n

Requires ALTPRI privilege to set the priority higher than your base priority. Changes the priority for the specified process. If you do not have the ALTPRI privilege, the value you specify is compared to your current base priority, and the lower value is always used.

/PRIVILEGES=(privilege[,...])

Requires SETPRV privilege to enable a privilege you do not have. Enables privileges for the process. For a list of process privileges, see Table 4-1 in the *VMS System Manager's Manual*. Use the SHOW PROCESS /PRIVILEGES command to determine what privileges are currently enabled.

/RESOURCE_WAIT
/NORESOURCE_WAIT

Enables resource wait mode so that the process waits for resources to become available. If you specify the /NORESOURCE_WAIT qualifier, the process receives an error status code when system dynamic memory is not available or when the process exceeds one of the following resource quotas: direct I/O limit, buffered I/O limit, or buffered I/O byte count (buffer space) quota.

/RESUME

Allows a process suspended by a previous SET PROCESS command to resume operation.

/SUSPEND[=SUPERVISOR]
/SUSPEND=KERNEL
/NOSUSPEND

Requires privileges as described in text. Temporarily stops the process's activities. The process remains suspended until another process resumes or deletes it. The qualifiers /NOSUSPEND and /RESUME allow a suspended process to resume operation.

Specify either of the following keywords with /SUSPEND to produce different results:

Keyword	Result
SUPERVISOR (default)	Specifies that the named process is to be suspended to allow the delivery of Asynchronous System Traps (ASTs) at EXEC or KERNEL mode. Specifying this keyword is optional.
KERNEL	Specifies that the named process is to be suspended such that no asynchronous system traps (ASTs) can be delivered. To specify the KERNEL keyword, you must be in either kernel mode or exec mode, or have either CMKRNL or CMEXEC privilege enabled. Note that this was the default behavior of SET PROCESS/SUSPEND for versions of VMS prior to Version 5.0.

Depending on the operation, the process from which you specify /SUSPEND requires privileges. You must have GROUP privilege to suspend another process in the same group, unless that process has the same UIC. You must have WORLD privilege to suspend any other process in the system.

Note that you can specify SET PROCESS/SUSPEND=KERNEL to override a previous SET PROCESS/SUSPEND=SUPERVISOR. SET PROCESS /SUSPEND=SUPERVISOR does not, however, override SET PROCESS /SUSPEND=KERNEL.

/SWAPPING (default)
/NOSWAPPING

Requires the user privilege PSWAPM to disable swapping for your process. Permits the process to be swapped.

example

```
$ RUN/PROCESS_NAME=TESTER  CALC
%RUN-S-PROC_ID, identification of created process is 0005002F
$ SET PROCESS/PRIORITY=10  TESTER
```

The RUN command in this example creates a subprocess and gives it the name TESTER. Subsequently, the SET PROCESS/PRIORITY command assigns the subprocess a priority of 10.

SET PROMPT

Replaces the default DCL prompt ($) with the specified string.

format

SET PROMPT*[=string]*

parameter

string

Specifies the new prompt string. The following rules apply:

- All valid ASCII characters can be used.
- No more than 32 characters are allowed.
- To include spaces or lowercase letters, enclose the string in quotation marks. Otherwise, letters are automatically converted to uppercase; leading and trailing spaces are removed.

If you do not specify the string parameter with the SET PROMPT command, the default DCL prompt ($) is restored.

qualifier

/CARRIAGE_CONTROL (default)
/NOCARRIAGE_CONTROL

Inserts carriage return and line feed characters before the prompt string. Type the qualifier after the string parameter.

example

```
$ SET PROMPT ="What's next?"
What's next? SHOW TIME
 31-DEC-1988 14:08:58
```

The SET PROMPT command in this example replaces the DCL prompt ($) with the phrase "What's next?". When you see the prompt on your screen, you can enter any DCL command. This example uses the SHOW TIME command.

SET PROTECTION

Establishes the protection that limits other users' access to a file or a group of files.

You cannot change the protection on a file on a network node other than the one you are currently logged in to.

format

SET PROTECTION*[=(code)] file-spec[,...]*

parameters

code
Defines the protection to be applied to the specified files. If you omit the code, the access is set to the current default protection.

file-spec[,...]
Specifies one or more files for which the protection is to be changed. A file name and file type are required. If you omit a version number, the protection is changed only for the highest existing version of the file. Wildcard characters are allowed.

qualifiers

/CONFIRM
/NOCONFIRM (default)
Controls whether the SET PROTECTION command displays the file specification of each file before applying the new protection, and requests you to confirm that the file's protection should be changed. To change the protection, type Y (YES) or T (TRUE) at the system prompt and press RETURN. If you enter anything else, such as N or NO, the file protection is not changed.

/LOG
/NOLOG (default)
Controls whether the system displays the file specification of each file for which the protection is changed as the command executes.

/PROTECTION=(code)
File-spec qualifier. If you follow a file specification with the /PROTECTION qualifier, the code specified with /PROTECTION overrides the command's code parameter. The /PROTECTION qualifier lets you assign different protection codes to several files with a single SET PROTECTION command.

example

```
$ SET PROTECTION A.DAT, B.DAT/PROTECTION=OWNER:RWED, C.DAT
```

The SET PROTECTION command in this example specifies that the file A.DAT receive the default protection established for your files. The existing protection for the file B.DAT is overridden, only for the owner category, to provide read, write, execute, and delete access. Note that no protection is specified for the file C.DAT at either the command or file level. Like A.DAT, C.DAT receives the default protection.

Since no version numbers are specified, the protection settings affect only the highest versions of the three files.

SET PROTECTION/DEFAULT

Establishes the default protection to be applied to all files subsequently created.

format

SET PROTECTION*[=(code)]/DEFAULT*

parameter

code

Defines the default protection to be applied to all files. To override this default protection use either the SET PROTECTION or CREATE commands. If you do not specify a protection code, the current default protection remains unchanged.

example

```
$ SET PROTECTION=(GROUP:RWED,WORLD:R)/DEFAULT
```

The SET PROTECTION/DEFAULT command in this example sets the default protection to grant unlimited access to other users in the same group and read access to all users. The default protections for system and owner are not changed.

SET PROTECTION/DEVICE

Establishes the protection to be applied to a specific non-file-structured device. The protection for a device limits the type of access available to users. The /DEVICE qualifier is required.

Requires OPER privilege.

format

SET PROTECTION=*(ownership[:access],...)*/**DEVICE**
device-name[:]

parameters

ownership
An ownership category—SYSTEM, OWNER, GROUP, or WORLD. Each category can be abbreviated to its first character. Any protection code category that the operator does not specify will remain unchanged.

access
An access category—R (READ), W (WRITE), L (LOGICAL I/O), and P (PHYSICAL I/O)—to be assigned to a specified type of owner. A null access specification means no access.

device-name[:]
Specifies the name of the non file-structured device whose protection is to be set or modified.

qualifier

/OWNER_UIC=uic
Requests that the specified user identification code (UIC) be assigned ownership of the device for the purpose of access checks. The default owner is the UIC of the process entering the SET PROTECTION command.

example

```
$ SET PROTECTION=(S:RWLP,O:RWLP,G,W)/DEVICE  LAA0:
```

The command in this example requests that the protection for device LAA0 be set to allow all types of access to system processes and processes with the UIC of the current process. This command also denies access to anyone else.

SET QUEUE

Changes the current status or attributes of the specified queue.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

SET QUEUE *queue-name[:]*

parameter

queue-name[:]
Specifies the name of an execution queue or a generic queue.

qualifiers

/BASE_PRIORITY=n
Specifies the base process priority at which jobs are initiated from a batch queue. (You must stop and restart symbiont queues to change the symbiont priority for printer, terminal, or server queues.) The base priority can be any decimal value from 0 through 15.

/BLOCK_LIMIT=([lowlim,]uplim)
/NOBLOCK_LIMIT (default)
Limits the size of print jobs that can be executed on a printer or terminal queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters. The lower parameter specifies the minimum number of blocks accepted by the queue for a print job. The upper parameter specifies the maximum number of blocks accepted by the queue for a print job. If a job contains fewer blocks than the number specified by the lower parameter or more blocks than the number specified by the upper parameter, the job remains pending until the block limit for the queue is changed, enabling the job to execute.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS
Specifies one or more characteristics for processing jobs on the queue. If only one characteristic is specified, you can omit the parentheses. Each time you specify /CHARACTERISTICS, all previously set characteristics are erased. Only the ones specified with the qualifier are now established for the queue. Queue characteristics are installation specific. The characteristic parameter can be either a value from 0 through 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.

/CLOSE
Prevents jobs from being entered in the queue through PRINT or SUBMIT commands or as a result of requeue operations. To allow jobs to be entered, use the /OPEN qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled). When

a queue is marked closed, jobs executing continue to execute and jobs already pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Indicates the default CPU time limit for batch jobs. Time can be specified as delta time, 0, NONE (default), or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time (subject to the restrictions imposed by the /CPUMAXIMUM qualifier or the user authorization file). The keyword NONE specifies that no time limit is needed. The time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier. See *VMS General User's Manual* for information on specifying delta time.

/CPUMAXIMUM=time

Indicates the maximum CPU time limit for batch jobs. The /CPUMAXIMUM qualifier overrides the time limit specified in the user authorization file (UAF). Time can be specified as delta time, 0, NONE (default), or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time (subject to the restrictions imposed by the /CPUMAXIMUM qualifier or the user authorization file). The keyword NONE specifies that no time limit is needed.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. Once an option is set for the queue by the /DEFAULT qualifier, users do not have to specify that option in their PRINT commands. Possible options are as follows:

[NO]BURST[=keyword]	Specifies whether to print burst pages (flag pages printed over the paper's perforations for easy identification of individual files in a print job). The keyword ALL places burst pages before each printed file in the job. The keyword ONE places a burst page before the first printed file in the job.
[NO]FEED	Specifies whether a form feed is automatically inserted at the end of a page.
[NO]FLAG[=keyword]	Specifies whether to print flag pages. The keyword ALL places flag pages before each printed file in the job. The keyword ONE places a flag page before the first printed file in the job.

FORM=type	Specifies the default form for a printer, terminal, or server queue. If a job is not submitted with an explicit form definition, this form is used to process the job. The systemwide default form, form=0, is the default value for this keyword. See also /FORM_MOUNTED.
[NO]TRAILER[=keyword]	Specifies whether to print trailer pages. The keyword ALL places trailer pages after each printed file in the job. The keyword ONE places a trailer page after the last printed file in the job.

/DESCRIPTION=string
/NODESCRIPTION (default)

A string of up to 255 characters used to provide operator-supplied information about the queue.

If the string contains alphanumeric, underscore, or dollar sign characters it must be enclosed in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may have been associated with the queue.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING (default)

Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC (default)
/NOENABLE_GENERIC

Specifies whether files queued to a generic queue that does not have specific targets can be placed in this execution queue for processing.

/FORM_MOUNTED=type

Specifies the form type for a printer, terminal, or server queue. If the stock of the mounted form is not identical to the stock of the default form, as indicated by the DCL command qualifier /DEFAULT=FORM=type, all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, then the job enters a pending state. In both cases, jobs remain pending until the stock of the mounted form of the queue is identical to the stock of the form associated with the job. To specify the form type, use a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are specific to each installation.

/JOB_LIMIT=n

Indicates the number of batch jobs that can be executed concurrently from the queue.

/OPEN

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled).

/OWNER_UIC=uic

Requires OPER privilege. Enables you to change the user identification code (UIC) of the queue.

/PROTECTION=(ownership[:access],...)

Requires OPER privilege. Specifies the protection of the queue. By default, the queue protection is (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W). If you include only one protection code, you can omit the parentheses.

/RECORD_BLOCKING
/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify /NORECORD_BLOCKING, the symbiont is directed to send each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN

Retains jobs in the queue in a completed status after they have executed. Possible options are as follows:

ALL	Retains all jobs in the queue after execution (default)
ERROR	Retains in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in a printer or terminal queue are scheduled for printing based on the size of the job. When /SCHEDULE=SIZE (the default) is in effect, shorter jobs print before longer ones. With /SCHEDULE=NOSIZE, jobs are printed in the order they were submitted, regardless of size.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEPARATE=(option[,...])
/NOSEPARATE

Specifies the job separation defaults for a printer or terminal queue. The job separation options are as follows:

[NO]BURST	Specifies whether a burst page prints at the beginning of every job. Specifying BURST also results in a flag page being printed.
[NO]FLAG	Specifies whether a flag page prints at the beginning of every job.
[NO]TRAILER	Specifies whether a trailer page prints at the end of every job.
[NO]RESET=(module[,...])	Specifies a job reset sequence for the queue. The specified modules from the device control library are used to reset the device each time a job reset occurs.

/WSDEFAULT=n

Defines a working set default for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Specify a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set default value defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

/WSEXTENT=n

Defines a working set extent for the batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Specify a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set extent value defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

/WSQUOTA=n

Defines the working set page size (working set quota) for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Specify a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set quota value defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

example

```
$ INITIALIZE/QUEUE/DEFAULT=BURST/FORM_MOUNTED=LETTER/START SYS$PRINT
.
.
.
$ STOP/QUEUE/NEXT SYS$PRINT
$ SET QUEUE /DEFAULT=BURST/FORM_MOUNTED=MEMO SYS$PRINT
```

In this example, the queue is initialized with the INITIALIZE/QUEUE command to have the following attributes: a burst page preceding each file in the job and the form type LETTER. Later the queue is stopped with the STOP/QUEUE/NEXT command so that the current job finishes processing before the queue stops. The SET QUEUE command changes the form type to MEMO.

SET QUEUE/ENTRY

Changes the current status or attributes of a job that is not currently executing in a queue. The /ENTRY qualifier is required.

As of VMS Version 5.0, the SET QUEUE/ENTRY command is superseded by the SET ENTRY command. Note that the SET ENTRY command has the same qualifiers as the SET QUEUE/ENTRY command; only the command parameters are different. DIGITAL recommends usage of the SET ENTRY command.

Requires OPER privilege or EXECUTE (E) access to the specified queue. If you have DELETE (D) access to the specified job, you can alter the attributes for that job. In addition, the queue name parameter is optional.

format

SET QUEUE/ENTRY=*entry-number queue-name[:]*

SET RESTART_VALUE

Sets a value for the symbol BATCH$RESTART, used for restarting portions of batch jobs. If the system encounters the command interactively, no action is taken. Use the SET RESTART_VALUE command in command procedures.

format

SET RESTART_VALUE=*string*

parameter

string
Specifies a string of up to 255 characters specifying the label at which the batch job should begin executing again.

example

```
$ SET RESTART_VALUE=FIRST_PART
```

The SET RESTART_VALUE command in this example sets the value of BATCH$RESTART to FIRST_PART.

SET RIGHTS_LIST

Allows users to modify the process or system rights list. You must specify either /DISABLE or /ENABLE with the SET RIGHTS_LIST command.

format

SET RIGHTS_LIST *id-name[,...]*

parameter

id-name[,...]

Specifies identifiers to be added to or removed from the process or system rights list. The id-name parameter is a string of 1 to 31 alphanumeric characters, underscores, and dollar signs; each name must contain at least one nonnumeric character.

qualifiers

/ATTRIBUTES=(keyword[,...])

Specifies attributes to be associated with the identifiers. Attributes may be added to new or existing identifiers. The following are valid keywords:

[NO]DYNAMIC	Indicates whether or not unprivileged holders of the identifiers may add or remove them from the process rights list. The default is NODYNAMIC.
[NO]RESOURCE	Indicates whether or not holders of the identifiers may charge resources to them. The default is NORESOURCE.

/DISABLE

Removes the identifiers from the process or system rights list. You cannot use /DISABLE with the /ENABLE qualifier.

/ENABLE

Adds the identifiers to the process or system rights list. You cannot use /ENABLE with the /DISABLE qualifier.

/IDENTIFICATION=pid

Specifies the process identification value (PID) of the process whose rights list is to be modified. The PID is assigned by the system when the process is created. When you specify a PID, you can omit the leading zeros.

If you specify the /IDENTIFICATION qualifier, you cannot use the /PROCESS qualifier. By default, if neither the /IDENTIFICATION nor the /PROCESS qualifier is specified, the current process is assumed. You cannot use /IDENTIFICATION with the /SYSTEM qualifier.

/PROCESS[=process-name]

Specifies the name of the process whose rights list is to be modified. The process name can contain from 1 to 15 alphanumeric characters.

If you specify the /PROCESS qualifier, you cannot use the /IDENTIFICATION qualifier. By default, if neither the /PROCESS nor the /IDENTIFICATION qualifier is specified, the current process is assumed.

/SYSTEM

Specifies that the desired operation (addition or removal of an identifier) be performed on the system rights list. You cannot use /SYSTEM with /PROCESS or /IDENTIFICATION.

example

```
$ SET RIGHTS_LIST/ENABLE/ATTRIBUTES=RESOURCE MARKETING
```

The SET RIGHTS_LIST command in this example adds the MARKETING identifier to the process rights list of the current process. Specifying the RESOURCE attribute allows holders of the MARKETING identifier to charge resources to it.

SET RMS_DEFAULT

Defines default values for the multiblock and multibuffer counts, network transfer sizes, prolog level, and extend quantity used by VMS RMS for file operations. If you set the default for either the multiblock count or the multibuffer count at 0, VMS RMS tries to use the process default value or the system default value, in that order. If these are set at 0, VMS RMS uses a default value of 1.

format

SET RMS_DEFAULT

parameters

None.

qualifiers

/BLOCK_COUNT=count

Specifies a default multiblock count (0 through 127) for record I/O operations *only*, where count is the number of blocks to be allocated for each I/O buffer.

/BUFFER_COUNT=count

Specifies a default multibuffer count (0 through 127) for file operations. If file type is not specified, the default is applied to sequential files.

/DISK

Applies the specified defaults to disk file operations.

/EXTEND_QUANTITY=n

Specifies the number of blocks (n) to extend a sequential file where n can range from 0 to 65535. If you do not specify /EXTEND_QUANTITY, VMS RMS calculates its own extend value. The /EXTEND_QUANTITY qualifier value is used when the program does not explicitly specify an extent quantity.

/INDEXED

Applies the multibuffer default to indexed file operations.

/MAGTAPE

Applies the multibuffer default to magnetic tape operations.

/NETWORK_BLOCK_COUNT=count

Specifies a default block count (0 through 127) for network access to remote files, where count represents the number of I/O buffers that VMS RMS allocates for transmitting and receiving data. If you omit the value or specify a value of 0, VMS RMS uses the systemwide block count value. If this value is also 0, VMS RMS uses a size of one block.

/PROLOG=n

Specifies a default prolog level for indexed sequential files where acceptable values for n are 0, 2 or 3. If 0 (default) is specified, VMS RMS sets an appropriate prolog level.

/RELATIVE

Applies the multibuffer default to relative file operations.

/SEQUENTIAL (default)

Applies the multibuffer default to sequential file operations.

/SYSTEM

Requires CMKRNL privilege. Applies specified defaults on a systemwide basis to all file operations.

/UNIT_RECORD

Applies the multibuffer default to file operations on unit record devices.

example

```
$ SET RMS_DEFAULT/BUFFER_COUNT=7/NETWORK_BLOCK_COUNT=16/SYSTEM
$ SHOW RMS_DEFAULT
```

	MULTI-BLOCK COUNT	MULTIBUFFER COUNTS					NETWORK BLOCK COUNT
		Indexed	Relative	Sequential			
				Disk	Magtape	Unit Record	
Process	24	0	0	0	8	0	0
System	16	0	0	7	7	7	16

	Prolog	Extend	Quantity
Process	0		0
System	0		0

The SET RMS_DEFAULT command in this example defines the systemwide default multibuffer count at 7 for all sequential file operations on disk, magnetic tape, and unit record devices. The command also sets the network block count at 16.

SET SYMBOL

Controls access to local and global symbols in command procedures.

format

SET SYMBOL

qualifier

/SCOPE=(keyword,...)

Controls access to local and global symbols. Lets you treat symbols as being undefined. Possible keywords are as follows:

NOLOCAL	Causes all local symbols defined in outer procedure levels to be treated as being undefined by the current procedure and all inner procedure levels.
LOCAL	Removes any symbol translation limit set by the current procedure level.
NOGLOBAL	Causes all global symbols to be inaccessible to the current procedure level and all inner procedure levels unless otherwise changed.
GLOBAL	Restores access to all global symbols.

example

```
$ SET SYMBOL/SCOPE=NOLOCAL
```

In this example, all local symbols defined in outer procedure levels are now undefined by the current procedure and all inner procedure levels.

SET TERMINAL

Sets the characteristics of a terminal. Entering a qualifier changes a characteristic; omitting a qualifier leaves the characteristic unchanged.

format

SET TERMINAL *[device-name[:]]*

parameter

device-name[:]
Specifies the device name of the terminal. The default is SYS$COMMAND if that device is a terminal. If the device is not a terminal, an error message is displayed.

qualifiers

/ADVANCED_VIDEO
/NOADVANCED_VIDEO
Specifies that the terminal has advanced video attributes and is capable of 132-column video. If the terminal width is set to 132 columns and /ADVANCED_VIDEO is enabled, the terminal page limit is set to 24 lines. If /NOADVANCED_VIDEO is enabled, the terminal page limit is set to 12 lines.

/ALTYPEAHD
Sets the size of the type-ahead buffer when used with the /PERMANENT qualifier. You should specify SET TERMINAL/PERMANENT/ALTYPEAHD in the SYS$SYSTEM:STARTUP.COM for those communication lines that require this capability.

To use this feature interactively, specify SET TERMINAL/PERMANENT /ALTYPEAHD. This specification is effective at your next login.

/ANSI_CRT (default)
/NOANSI_CRT
Specifies whether the terminal conforms to ANSI CRT programming standards. Since ANSI standards are a proper subset of the DEC_CRT characteristics, the default for all VT100-family terminals is /ANSI_CRT.

/APPLICATION_KEYPAD
Specifies that the keypad is to be set to APPLICATION_KEYPAD mode, which allows you to enter DCL commands defined with the DEFINE/KEY command. By default, the terminal is set to NUMERIC_KEYPAD mode.

/AUTOBAUD
/NOAUTOBAUD
Specifies whether the terminal baud rate is set when you log in and sets the default terminal speed to 9600. You must press the RETURN key two or more times at intervals of at least one second for the baud rate to be correctly

determined. If you press a key other than RETURN, /AUTOBAUD might detect the wrong baud rate. If this happens, wait for the login procedure to time out before continuing. The /AUTOBAUD qualifier must be used with the /PERMANENT qualifier.

The valid baud rates are as follows:

110	1200	4800
150	1800	9600
300	2400	19200
600	3600	

/BLOCK_MODE
/NOBLOCK_MODE
Performs block mode transmission, local editing, and field protection.

/BRDCSTMBX
/NOBRDCSTMBX
Sends broadcast messages to an associated mailbox if one exists.

/BROADCAST (default)
/NOBROADCAST
Enables reception of broadcast messages (such as those issued by MAIL and REPLY). Specify the /NOBROADCAST qualifier when you are using a terminal as a noninteractive device or when you do not want special output to be interrupted by messages. Use SET BROADCAST to exclude certain types of messages from being broadcast, rather than eliminating all messages.

/CRFILL[=fill-count]
Generates the specified number of null characters after each carriage return before transmitting the next meaningful character (to ensure that the terminal is ready for reception). The value must be an integer in the range 0 through 9. The default is /CRFILL=0.

/DEC_CRT[=(value1,value2,value3)]
/NODEC_CRT[=(value1,value2,value3)]
Specifies that the terminal conforms to DIGITAL VT100-, VT200-, or VT300-family standards and supports the minimum standards, including the additional DIGITAL escape sequences.

One of the following three optional values may be specified:

1 (default)	Requests that the DEC_CRT terminal characteristic be set.
2	Requests that the DEC_CRT2 terminal characteristic be set.
3	Requests that the DEC_CRT3 terminal characteristic be set. A level 3 terminal is described as follows:

- Supports a status line (line 25, at the bottom of the screen)
- Supports the IOS Latin-1 character set
- Has terminal state interrogation (describes what state your terminal is in)

/DEVICE_TYPE=terminal-type

Informs the system of the terminal type and sets characteristics according to the device type specified. You can specify any of the following terminal types:

UNKNOWN	LA34
FT1 - FT8	LA38
LA12	LA100
LA36	LQP02
LA120	VT125
LN03	LN01K
VT05	VT131
VT52	VT132
VT55	VT173
VT100	VT200
VT101	PRO_SERIES
VT102	LA210
VT105	VT300

The default characteristics for the VT100, VT102, and VT125 series terminals are as follows:

/ADVANCEDVIDEO	/CRFILL=0	/LFFILL=0	/SPEED=9600
/NOALTYPEAHD[1]	/ECHO	/LOWERCASE	/TAB
/ANSI_CRT	/NOEIGHT_BIT	/NODMA	/TTSYNC
/NOAUTOBAUD	/NOESCAPE	/PAGE=24	/TYPE_AHEAD
/NOBLOCK_MODE	/NOFORM	/NOPARITY	/WIDTH=80
/NOBRDCSTMBX	/FULLDUP	/NOPASTHRU	/WRAP
/BROADCAST	/NOHOSTSYNC	/NOREADSYN	

[1]This is the default characteristic set by the system and is not a valid qualifier for your use.

/DIALUP
/NODIALUP (default)
Specifies that the terminal is a dial-up terminal.

/DISCONNECT
/NODISCONNECT (default)
Specifies that the process connected to this terminal not be disconnected if the line detects a hangup. The /DISCONNECT qualifier is valid only when /PERMANENT is specified.

/DISMISS
/NODISMISS (default)
Causes the terminal driver to ignore data causing a parity error (instead of terminating the currently outstanding I/O with an error status).

/DMA
/NODMA
Controls the use of direct memory access (DMA) mode on a controller that supports this feature.

/ECHO (default)
/NOECHO
Causes the terminal to display the input it receives. With /NOECHO, the terminal displays only system or user application output, or both.

/EDIT_MODE
/NOEDIT_MODE
Specifies that the terminal can perform ANSI-defined advanced editing functions.

/EIGHT_BIT
/NOEIGHT_BIT
Uses 8-bit ASCII protocol rather than 7-bit ASCII protocol.

/ESCAPE
/NOESCAPE (default)
Validates escape sequences.

/FALLBACK
/NOFALLBACK
Displays the 8-bit DEC Multinational Character Set characters on the terminal in their 7-bit representation. The default depends on the /EIGHTBIT setting of the terminal.

/FORM
/NOFORM
Transmits a form feed rather than translating it into multiple line feeds.

/FRAME=n

Specifies the number of data bits that the terminal driver expects for every character that is input or output. The value of *n* can be from 5 through 8. The default value depends on the /PARITY and /EIGHTBIT settings of the terminal.

/FULLDUP (default)
/NOFULLDUP

Operates in full duplex mode. The /FULLDUP qualifier is equivalent to /NOHALFDUP.

/HALFDUP
/NOHALFDUP (default)

Operates in half duplex mode. The /HALFDUP qualifier is equivalent to /NOFULLDUP.

/HANGUP
/NOHANGUP (default)

May require LOG_IO or PHY_IO privilege depending on system generation parameter settings. Controls whether the terminal modem is hung up when you log out.

/HARDCOPY
/NOHARDCOPY

Establishes the device as a hardcopy terminal and outputs a backslash (\) when the DELETE key is pressed. The /HARDCOPY qualifier is equivalent to /NOSCOPE.

/HOSTSYNC
/NOHOSTSYNC (default)

When you specify the /HOSTSYNC qualifier, the system stops transmission to the terminal (by generating a CTRL/S) when the input buffer is full and resumes transmission (by generating a CTRL/Q) when the input buffer is empty.

/INQUIRE

Sets the device type according to a response elicited from the terminal; the default is UNKNOWN. Works only on DIGITAL terminals, and not on the LA36 or VT05 terminals. Some VT100-family terminals, including the VT101 and VT105, return a VT100-type response. LA38 terminals respond as LA43 terminals.

You can include the SET TERMINAL/INQUIRE command in your LOGIN.COM file to automatically detect the terminal type.

CAUTION: This qualifier clears the type-ahead buffer. If the response sequence is unrecognized, no action message or error message is displayed. The /INQUIRE qualifier should be used only on DIGITAL terminals. However, the LA36 and VT05 terminals do not support this feature.

/INSERT
Sets the terminal to /INSERT mode. This feature allows you to insert characters when editing command lines. The default mode is /OVERSTRIKE, which allows you to type over the current character when editing a command line. Use CTRL/A to switch from one mode to the other.

/LFFILL[=fill-count]
Transmits to the terminal the specified number of null characters after each line feed before transmitting the next meaningful character (to ensure that the terminal is ready for reception). The value must be an integer in the range 0 through 9.

/LINE_EDITING
/NOLINE_EDITING
Enables advanced line-editing features for editing command lines: both RETURN and CTRL/Z are recognized as line terminators, as are escape sequences.

/LOCAL_ECHO
/NOLOCAL_ECHO (default)
Echoes characters locally (rather than the host echoing them) for command level terminal functions. (Do not use /LOCAL_ECHO with utilities that require control over echoing, such as line editing or EDT's screen mode.)

CAUTION: When logging in to terminals with /LOCAL_ECHO set, the VMS operating system has no control over the echoing of passwords.

/LOWERCASE
/NOLOWERCASE
Passes lowercase characters to the terminal. The /NOLOWERCASE qualifier translates all input to uppercase. /LOWERCASE is equivalent to /NOUPPERCASE.

/MANUAL
Indicates manual switching of terminal lines to dynamic asynchronous DDCMP lines when your local terminal emulator does not support automatic switching. The /MANUAL qualifier should be specified with the /PROTOCOL=DDCMP and /SWITCH=DECNET qualifiers.

/MODEM
/NOMODEM
Indicates that the terminal is connected to a modem or a cable that supplies standard EIA modem control signals. If your terminal has the MODEM characteristic, typing SET TERMINAL/NOMODEM automatically logs you out.

/SPEED=(input-rate,output-rate)
Sets the baud rate at which the terminal receives and transmits data. If the input and output rates are the same, specify /SPEED=rate.

Not all terminals support different input and output baud rates. For specific information on baud rates for your terminal, consult the manual for that terminal.

The default transmission rates are installation-dependent.

/SWITCH=DECNET
Causes the terminal lines at each node to be switched to dynamic asynchronous DDCMP lines, when specified with the /PROTOCOL=DDCMP qualifier. Note that /SWITCH=DECNET is a permanent characteristic; therefore, the /PERMANENT qualifier is not required.

/SYSPASSWORD
/NOSYSPASSWORD (default)
Requires LOG_IO privilege. Determines whether the terminal requires that a system password be entered before the *Username:* prompt.

/TAB
/NOTAB
Does not convert tab characters to multiple blanks. The /NOTAB qualifier expands all tab characters to blanks and assumes tab stops at 8-character intervals.

/TTSYNC (default)
/NOTTSYNC
Stops transmitting to the terminal when CTRL/S is pressed and resumes transmission when CTRL/Q is pressed.

/TYPE_AHEAD (default)
/NOTYPE_AHEAD
Accepts unsolicited input for the terminal to the limit of the type-ahead buffer.

When you specify /NOTYPE_AHEAD, the terminal is dedicated, and accepts input only when a program or the system issues a read to the terminal. Logins are disabled on a terminal with /NOTYPE_AHEAD set. When you specify /TYPE_AHEAD, the amount of data that can be accepted is governed by the size of the type-ahead buffer. That size is determined by system generation parameters.

/UNKNOWN
Specifies a terminal type that is unknown to the system, which then uses the default terminal characteristics for unknown terminals.

/UPPERCASE
/NOUPPERCASE
Translates lowercase to uppercase characters. The /UPPERCASE qualifier is equivalent to /NOLOWERCASE.

/WIDTH=characters-per-line
Specifies the maximum characters per line. This value must be an integer in the range 1 through 511. With /WRAP, the terminal generates a carriage return and line feed when the width specification is reached.

If the specified width on an ANSI terminal is 132, the screen is set to 132-character mode. If the terminal does not have advanced video option (AVO), the page length limit is set to 12 lines.

/WRAP (default)
/NOWRAP
Generates a carriage return and line feed when the value of /WIDTH is reached.

example

```
$ SET TERMINAL/WIDTH=132/PAGE=60/NOBROADCAST
$ TYPE MEMO.DOC
.
.
.
$ SET TERMINAL/DEVICE=LA36
```

In this example, the first SET TERMINAL command indicates that the width of terminal lines is 132 characters and that the size of each page is 60 lines. The /NOBROADCAST qualifier disables the reception of broadcast messages while the terminal is printing the file MEMO.DOC. The next SET TERMINAL command restores the terminal to its default state.

SET TIME

Resets the system clock, which is used both as a timer to record intervals between various internal events, and as a source clock for displaying the time of day.

Requires both OPER and LOG_IO privileges.

format

SET TIME*[=time]*

parameter

time

Specifies a date in the format day-month-year, or a time in the format hour:minute:second.hundredth, or both. Day must be an integer in the range 1 through 31. Month must be JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC. Year must be an integer in the range 1858 through 9999. Hour must be an integer in the range 0 through 23. Minute must be an integer in the range 0 through 59. Second must be an integer in the range 0 through 59. Hundredth (of a second) must be an integer in the range 0 through 99. The hyphens, colons, and period are required delimiters. Delimit the date and time, when both are specified, with a colon.

example

```
$ SET TIME=31-DEC-1988:19:31:0.0
```

The SET TIME command in this example sets the date/time at December 31, 1988, 7:31 p.m.

SET UIC

Changes the user identification code (UIC) of your process.

Requires CMKRNL privilege.

format

SET UIC *uic*

parameter

uic

Specifies a valid UIC. Brackets are required around the UIC.

example

```
$ SET UIC [370,10]
```

The SET UIC command in this example establishes your UIC as [370,10]. You can now read or modify any files whose access is restricted to this UIC.

SET VERIFY

Controls whether command lines and data lines in command procedures are displayed at the terminal or printed in a batch job log. The information displayed by the SET VERIFY command can help you in debugging command procedures.

format

SET [NO]VERIFY*[=([NO]PROCEDURE, [NO]IMAGE)]*

parameter

([NO]PROCEDURE, [NO]IMAGE)

Specifies one or both types of verification. Procedure verification causes each DCL command line in a command procedure to be written to the output device. Image verification causes data lines (input data that is included as part of the SYS$INPUT input stream) to be written to the output device. By default, both types of verification are set or cleared with SET VERIFY and SET NOVERIFY. If you specify only one keyword, the other is not affected. If you specify only one keyword, omit the parentheses.

description

By default, the SET VERIFY and SET NOVERIFY commands set or clear both types of verification. The default setting for command procedures executed interactively is SET NOVERIFY. System responses and error messages are, however, always displayed. The default for batch jobs is SET VERIFY.

example

```
$ PROC_VER = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ IMAGE_VER = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
    .
    .
    .
$ TEMP = F$VERIFY(PROC_VER, IMAGE_VER)
```

This command procedure uses the lexical function F$ENVIRONMENT to save the current procedure and image verification setting. Then the SET NOVERIFY command turns off both procedure and image verification. Subsequently, the F$VERIFY function is used to restore the original verification settings.

SET VOLUME

Changes the characteristics of one or more mounted Files-11 volumes.

Requires WRITE (W) access to the index file on the volume. If you are not the owner of the volume, requires either a system UIC or SYSPRV privilege.

format

SET VOLUME *device-name[:][,...]*

parameter

device-name[:][,...]
Specifies the name of one or more mounted Files-11 volumes.

qualifiers

/ACCESSED[=n]
Requires OPER privilege. Specifies the number of directories to be maintained in system space for ready access. You can specify a number (n) in the range of 0 through 255. If you specify the qualifier /ACCESSED and omit the number of directories, a default value of 3 is used.

/DATA_CHECK[=(option[,...])]
Defines a default for data check operations following all reads and writes to the specified volume. (If you do not specify the /DATA_CHECK qualifier, no checks are made.) Possible keywords are as follows:

READ	Performs checks following all read operations
WRITE	Performs checks following all write operations (default)

/ERASE_ON_DELETE
/NOERASE_ON_DELETE (default)
Determines whether the space occupied by a file is overwritten with a system specified pattern when a file on the volume is deleted.

/EXTENSION[=n]
Specifies the number of blocks to be used as a default extension size for all files on the volume. You can specify a number (n) in the range of 0 through 65,535. If you specify the /EXTENSION qualifier without specifying a value, a default value of 0 (the VMS RMS default) is used.

/FILE_PROTECTION=(code)
Sets the default protection to be applied to all files on the specified disk volume. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (READ), W (WRITE), E (EXECUTE), or D (DELETE). A null access specification means no access.

/HIGHWATER_MARKING
/NOHIGHWATER_MARKING
Determines whether the File Highwater Mark (FHM) volume attribute is set. The FHM attribute guarantees that a user cannot read data that was not written by the user. Applies to Structure Level 2 volumes only.

/LABEL=volume-label
Specifies a 1- through 12-character alphanumeric name to be encoded on the volume. Characters are automatically changed to uppercase.

/LOG
/NOLOG (default)
Determines whether the volume specification of each volume is displayed after the modification.

/MOUNT_VERIFICATION
/NOMOUNT_VERIFICATION
Determines whether mount verification is enabled. Mount verification prevents interruption to user input/output operations and notifies the operator of problems with the disk.

/OWNER_UIC[=uic]
Sets the owner UIC of the volume to the specified UIC. The default UIC is that of the current process. Brackets are required around the UIC.

/PROTECTION=(code)
Specifies the protection to be applied to the volume. The ownership categories are SYSTEM, OWNER, GROUP, and WORLD; the access categories are R (READ), W (WRITE), E (EXECUTE), and D (DELETE). The default protection is all types of access by all categories of user.

/REBUILD
Recovers caching limits for a volume that was improperly dismounted. If a disk volume was dismounted improperly (such as during a system failure), and was then remounted with the MOUNT/NOREBUILD command, you can use SET VOLUME/REBUILD to recover the caching that was in effect at the time of the dismount.

/RETENTION=(min[,max])
Specifies the minimum and maximum retention times to be used by the file system to determine the expiration date for files on the volume. When a file is created, its expiration date is set to the current time + maximum. Each time the file is accessed, the current time is added to the minimum time. If the sum is greater than the expiration date, a new expiration date is computed.

/UNLOAD (default)
/NOUNLOAD
Specifies whether the volume is unloaded (spun down) when the DCL command DISMOUNT is entered.

/USER_NAME[=user-name]

Specifies a user name of up to 12 alphanumeric characters to be recorded on the volume. The default name is the current process user name.

/WINDOWS[=n]

Specifies the number of mapping pointers to be allocated for file windows. The value of *n* can be from 7 through 80; the default value is 7.

example

```
$ SET VOLUME/ACCESSED=25/USER_NAME=MANAGER/LOG DBA0:
```

The SET VOLUME command in this example specifies that 25 directories are to be maintained in system space for ready access for the volume DBA0. The command also assigns the user name MANAGER to the volume and displays the volume specification after the volume is modified.

SET WORKING_SET

Redefines the default working set size for the process, or sets an upper limit to which the working set size can be changed by an image that the process executes. Working set limits cannot be set to exceed those defined in the user authorization file (UAF).

format

SET WORKING_SET

qualifiers

/ADJUST (default)
/NOADJUST

Enables or disables the system's changing of the process working set.

/EXTENT=n

Specifies the maximum number of pages that can be resident in the working set during image execution.

The extent value must be greater than the minimum working set defined at system generation, and it must be less than or equal to the authorized extent defined in the user authorization file.

If you specify a value greater than the authorized extent, the command sets the working set limit at the maximum authorized value.

/LIMIT=n

Specifies the size to which the working set is to be reduced at image exit.

If you specify a value greater than the current quota, the quota value is also increased.

/LOG
/NOLOG (default)
Determines whether or not confirmation of the SET WORKING_SET command is displayed.

/QUOTA=n
Specifies the maximum number of pages that any image executing in the process context can request. An image can set the working set size for the process by calling the Adjust Working Set Limit ($ADJWSL) system service.

example

```
$ SHOW WORKING_SET
  Working Set       /Limit=  150  /Quota=  700            /Extent= 700
  Adjustment enabled    Authorized Quota=  700  Authorized Extent= 700
$ SET WORKING_SET/QUOTA=1000
%SET-I-NEWLIMS, new working set:   Limit = 150  Quota = 700  Extent = 700
```

The SHOW WORKING_SET command in this example displays the current limit, quota, and extent, as well as the authorized quota and authorized extent. The SET WORKING_SET command attempts to set a quota limiting the maximum number of pages any image can request that is greater than the authorized quota. Note from the response that the quota was not increased.

SHOW ACCOUNTING

Displays the activities for which accounting is currently enabled. For more information about the Accounting Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

format

SHOW ACCOUNTING

parameters

None.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT
Specifies the file to which the display is written; by default, the display is written to the current SYS$OUTPUT device.

example

```
$ SHOW ACCOUNTING/OUTPUT=ACCOUNTING.SET
```

The SHOW ACCOUNTING command in this example writes the current setting of SET ACCOUNTING to the file ACCOUNTING.SET.

SHOW ACL

Allows you to display the access control list (ACL) of an object.

format

SHOW ACL *object-name*

parameter

object-name

Specifies the name of the object whose ACL is to be displayed. No wildcard characters are allowed in the object-name specification.

qualifier

/OBJECT_TYPE=type

Defines the object type of the object whose ACL is to be displayed. The following keywords are used to specify the object type:

FILE (default)	The object is a Files-11 disk file.
DEVICE	The object is a device.
SYSTEM_GLOBAL_SECTION	The object is a system global section.
GROUP_GLOBAL_SECTION	The object is a group global section.
QUEUE	The object is a batch or device (terminal, server, or printer) queue.
LOGICAL_NAME_TABLE	The object is a system logical name table.

example

```
$ SHOW ACL/OBJECT_TYPE=DEVICE TTA1
Object type: device,    Object name: VTA1
(IDENTIFIER=[SALES,FRANK],ACCESS=READ)
(IDENTIFIER=[123,321]+NETWORK,ACCESS=NONE)
.
.
.
```

The SHOW ACL command in this example displays the ACL of the device TTA1.

SHOW AUDIT

Displays the security auditing features that are enabled and the events that they report. Also identifies the security auditing failure mode in effect on the system.

Requires the SECURITY privilege.

format

SHOW AUDIT

qualifiers

/ALL (default)
Displays all available auditing information.

/FAILURE_MODE
Displays the failure mode currently in effect on the system.

/OUTPUT[=file-spec]
/NOOUTPUT
Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

example

```
$ SHOW AUDIT
Security alarm failure mode is set to:
    WAIT        Processes will wait for resource

Security alarms currently enabled for:
    BREAKIN:      (DIALUP,LOCAL,REMOTE,NETWORK,DETACHED)
    LOGIN:        (DIALUP)
    LOGOUT:       (DIALUP)
```

The SHOW AUDIT command in this example reveals that the terminals enabled as security operators will receive an alarm whenever the system detects a possible breakin attempt, a dialup at login time, or whenever a dialup connection logs out.

/NUMERIC_KEYPAD (default)
Specifies that the keypad is to be set to /NUMERIC_KEYPAD mode, which allows you to use the keys on the numeric keypad to type numbers and punctuation marks. In order to use the DEFINE/KEY facility, which allows you to enter DCL commands defined with the DEFINE/KEY command, set the terminal to /APPLICATION_KEYPAD. Specifies whether the keys of the numeric keypad are used to type numbers and punctuation marks (/NUMERIC_KEYPAD) or to enter DCL commands defined with the DEFINE/KEY command (/APPLICATION_KEYPAD).

/OVERSTRIKE (default)
Sets the terminal to /OVERSTRIKE mode. This feature allows you to type over the current character when you are editing a command line. Set your terminal to /INSERT if you want to insert characters when editing command lines. Use CTRL/A to switch from one mode to the other.

/PAGE[=lines-per-page]
For hardcopy terminals, specifies the number of print lines between perforations. (When the terminal reads a form feed, it advances the paper to the next perforation.) The value of n can be from 0 through 255 and defaults to 0 (which treats a form feed as a line feed).

/PARITY[=option]
/NOPARITY (default)
Passes data with odd or even parity, where option equals ODD or EVEN. If you specify /PARITY without an option, the value defaults to EVEN.

/PASTHRU
/NOPASTHRU (default)
Passes all data (including tabs, carriage returns, line feeds, and control characters) to an application program as binary data. The setting of /TTSYNC is allowed.

/PERMANENT
Requires LOG_IO or PHY_IO privilege. Sets characteristics on a permanent basis, that is, over terminal sessions. However, the characteristics revert to their initial values if the system is halted and restarted. Use in a system startup file to establish characteristics for all terminals on the system.

/PRINTER_PORT
/NOPRINTER_PORT
Specifies that the terminal has a printer port (an attribute not set by the SET TERMINAL/INQUIRE command).

/PROTOCOL=DDCMP
/PROTOCOL=NONE (default)
Controls whether the terminal port specified is changed into an asynchronous DDCMP line. The /PROTOCOL=NONE qualifier changes an asynchronous DDCMP line back into a terminal line. Note that /PROTOCOL=DDCMP

is a permanent characteristic; therefore, the /PERMANENT qualifier is not required.

/READSYNC
/NOREADSYNC (default)
Uses the CTRL/S and CTRL/Q functions to synchronize data transmitted from the terminal.

The default is /NOREADSYNC; the system does not use CTRL/S and CTRL/Q to control reads to the terminal. The /READSYNC qualifier is useful for certain classes of terminals that demand synchronization or for special-purpose terminal lines where data synchronization is appropriate.

/REGIS
/NOREGIS
Specifies that the terminal understands REGIS graphic commands.

/SCOPE
/NOSCOPE
Establishes the device as a video terminal. /SCOPE is equivalent to /NOHARDCOPY.

/SECURE_SERVER
/NOSECURE_SERVER (default)
Causes the BREAK key on the terminal to log out the current process (except on a virtual terminal). With /SECURE_SERVER in effect, pressing the BREAK key when there is no current process initiates the login sequence. With /NOSECURE_SERVER in effect, the break is ignored.

On terminals set with /AUTOBAUD, with the /SECURE_SERVER qualifier in effect, pressing the BREAK key disconnects the current process but is not required to start a new login sequence. However, when /NOAUTOBAUD is set, the /SECURE_SERVER characteristic requires a break to initiate a new login sequence.

/SET_SPEED
/NOSET_SPEED
Requires either LOG_IO or PHY_IO privilege. Allows the /SPEED qualifier to be used to change the terminal speed.

/SIXEL_GRAPHICS
/NOSIXEL_GRAPHICS
Specifies that the terminal is capable of displaying graphics using the sixel graphics protocol.

/SOFT_CHARACTERS
/NOSOFT_CHARACTERS
Specifies that the terminal is capable of loading a user-defined character set.

SHOW BROADCAST

Displays the message classes that are currently affected by the SET BROADCAST command.

format

SHOW BROADCAST

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

example

```
$ SHOW BROADCAST
Broadcasts are currently disabled for:
    MAIL
```

The SHOW BROADCAST display in this example indicates that SET BROADCAST=NOMAIL is in effect.

SHOW CLUSTER

Invokes the Show Cluster Utility (SHOW CLUSTER) to monitor and display cluster activity and performance.

format

SHOW CLUSTER

SHOW CPU

Displays the current state of the processors in a VMS multiprocessing system.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

SHOW CPU *[cpu-id,...]*

parameter

cpu-id
Decimal value representing the identity of a processor in a multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0.

description

The SHOW CPU command displays information about the status, characteristics, and capabilities of the processors active in and available to a VMS multiprocessing system.

You identify the processors to be displayed by using either the /ACTIVE qualifier, the /ALL qualifier, a CPU ID, or list of CPU IDs. If you specify none of these, the SHOW CPU command uses the /ALL qualifier by default.

You identify the type of information to be displayed by using the /BRIEF, /FULL, and /SUMMARY qualifiers. If you specify neither the /SUMMARY, /BRIEF, nor /FULL qualifier, SHOW CPU assumes the /BRIEF qualifier by default. However, if you likewise do not identify a processor or processors as the object of a command, SHOW CPU assumes a default of SHOW/ALL/SUMMARY.

qualifiers

/ACTIVE
Selects as the subject of the display only those processors that are members of the system's active set.

/ALL
Selects all configured processors, active and inactive, as the subject of the display.

/BRIEF
Produces information from the summary display and also lists the current CPU state and current process (if any) for each processor in the configuration.

/FULL
Produces information from the summary display. The /FULL qualifier also lists the current CPU state, current process (if any), revision levels, and capabilities for each configured processor. It indicates which processes can execute only on certain processors in the configuration. In addition, if one or more uniprocessing drivers are present in the system, the /FULL qualifier lists them by name.

/SUMMARY

Produces a display listing the processors in the VMS multiprocessing system, indicating which is the primary, which are configured, and which are active. The /SUMMARY qualifier also indicates the minimum revision levels required for processors in the system, which VMS synchronization image has been loaded into the operating system, and whether multiprocessing is enabled. If the presence of one or more uniprocessing drivers in the system prohibits the enabling of multiprocessing, the SHOW CPU command displays a warning message.

SHOW DEFAULT

Displays the current default device and directory.

format

SHOW DEFAULT

example

```
$ SHOW DEFAULT
  DISK1:[ALPHA]
$ SET DEFAULT DISK5:[HIGGINS.SOURCES]
$ SHOW DEFAULT
  DISK5:[HIGGINS.SOURCES]
```

The SHOW DEFAULT command in this example displays the current default device and directory names. The SET DEFAULT command changes these defaults, and the next SHOW DEFAULT command displays the new default device and directory.

SHOW DEVICES

Displays the status of a device on the system.

See the qualifier descriptions for restrictions.

format

SHOW DEVICES *[device-name[:]]*

parameter

device-name[:]

Specifies the name of a device for which information is to be displayed. You can specify a complete device name or only a portion of a device name.

qualifiers

/ALLOCATED
Displays all devices currently allocated to processes.

/BRIEF (default)
Displays brief information about the specified devices.

/FILES
Requires SYSPRV or BYPASS privileges to list read-protected files.
Displays a list of the names of all files open on a volume and their associated process name and process identification (PID). The specified device must be a mounted Files-11 volume. If the specified volume is a multivolume set, the files on each volume in the set are listed. If the /SYSTEM qualifier is also specified, only the names of installed files and files opened by the system are displayed. If the /NOSYSTEM qualifier is specified, only those files opened by processes are displayed.

/FULL
Displays a complete list of information about the devices.

/MOUNTED
Displays all devices that currently have volumes mounted on them.

/OUTPUT[=file-spec]
/NOOUTPUT
Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

/SYSTEM
/NOSYSTEM
Controls whether the names of installed files and files opened by the system are displayed.

/WINDOWS
Displays the window count and total size of all windows for files open on a volume. The file name and related process name and process identification (PID) are also displayed. The letter C in a display indicates that the file is open with "cathedral windows" (segmented windows).

example

```
$ SHOW DEVICES/FULL DMA0
Disk NODE1$DMA0:, device type RK07, is online, allocated, mounted,
error logging enabled
 Error count                    0  Operations completed            1257
 Owner UIC                  [1,4]  Owner process name             VANNOY
 Owner process ID       202000C8  Dev Prot  S:RWED,O:RWED,G:RWED,W:RWED
 Reference count                2  Default buffer size               512
 Volume label       JAKE_X239      Relative volume no.                 0
 Cluster size                   1  Transaction count                   2
 Free blocks                 3741  Maximum files allowed           13447
 Extend quantity                5  Mount count                         1
 Volume status            Process  ACP process name             DMA0BACP
 File ID cache size            64  Extent cache size                  64
 Quota cache size              64
Volume is subject to mount verification, file high-water marking
```

In this example, the SHOW DEVICES command requests a full listing of the status of the RK07 device DMA0. The device is located on NODE1 in a VAXcluster.

SHOW DEVICES/SERVED

Displays information on devices served by the MSCP server on this node. The /SERVED qualifier is required.

format

SHOW DEVICES/SERVED

description

The SHOW DEVICES/SERVED command displays information about the MSCP server and the devices it serves. This information is mostly used by system managers.

qualifiers

/ALL

This qualifier displays the information displayed by all of the qualifiers listed below except the /OUTPUT qualifier.

/COUNT

Displays the number of transfer operations completed, sorted by the size of the transfers, and the number of MSCP operations that have taken place since the MSCP server was started.

/HOST

Displays the names of the processors that have MSCP-served devices on line. SYSGEN's MSCP/HOST command determines how many hosts in the cluster can connect to the MSCP server at one time.

/OUTPUT=[filespec]

Redirects output from your terminal to the specified file. If you do not specify a file, or if you do not use this qualifier, output is sent to SYS$OUTPUT.

/RESOURCE

Displays information on the resources available to the MSCP server for use in processing I/O requests for the devices it serves. You make these resources available to the MSCP server when you use SYSGEN's MSCP command to start the MSCP server and use the qualifiers listed in the following table:

Qualifier	Item Specified
/BUFFER	The amount of buffer space available to the MSCP server
/FRACTION	The maximum size, in pages, of the buffer granted to an I/O request; for transfers of more data than will fit a buffer of the size specified by this qualifier, several CI transfers are needed
/SMALL	The minimum size, in pages, of the buffer that the MSCP server can grant to an I/O request; if less than this amount of buffer space is available, the I/O request must wait until at least this much buffer space becomes available; when this much space becomes available, the MSCP server grants the request a buffer
/PACKETS	The number of I/O-request packets (CDRPs) available to the MSCP server for processing I/O requests

example

```
$ SHOW DEVICES/SERVED
MSCP Served Devices on BOSTON 31-DEC-1988 12:34:56.78
                                        Queue Requests
Device:          Status     Total Size   Current    Max     Hosts
    2$DBA0        AVAIL        340670         0      0         0
    2$DMA1       ONLINE         53790         0      0         2
    2$DMA0      OFFLINE         53790         0      0         0
```

This example shows the output generated by the command SHOW DEVICES/SERVED. The first column in the display shows the names of the devices that are served by the MSCP server. The second column shows the status of the devices. The third column shows the size, in blocks, of the device.

The *Queue Requests* columns show the number of I/O requests currently awaiting processing by that device and the maximum number of I/O requests that have ever been concurrently awaiting processing by that device. The last column in the display shows the number of hosts that have the device on line.

SHOW ENTRY

Displays information about a user's batch and print jobs or about specific job entries.

Requires GROUP privilege to display all jobs in your group. Requires OPER privilege to display all jobs in all groups.

format

SHOW ENTRY *[entry-number,...]*

parameter

[entry-number,...]
Specifies the entry number of the job you want displayed. If no entry number is specified, all your jobs (or those owned by the user specified with the /USER_NAME qualifier) are displayed.

qualifiers

/BATCH
Selects batch jobs for display. If /USER_NAME is not specified, information about your jobs is displayed.

/BRIEF (default)
Displays the following information for each job: job name, user name, entry number, job size in blocks (for print jobs), status, queue name, and queue type. The /FULL and /FILES qualifiers override /BRIEF. Specify the /FULL qualifier to obtain more job information.

/BY_JOB_STATUS[=(keyword,...)]
Selects for display only those jobs with the specified status. Specify the status with one or more of the following keywords:

EXECUTING	Requests the display of currently executing jobs.
HOLDING	Requests the display of jobs on hold. Holding status indicates that the job is being held in the queue indefinitely.
PENDING	Requests the display of jobs with pending status. Pending status indicates that the job is waiting its turn to execute.
RETAINED	Requests the display of jobs retained in the queue after execution. Retained status indicates that the job has completed but remains in the queue. For example, a job may be retained in the queue if there was an error during its execution.
TIMED_RELEASE	Requests the display of jobs on hold until a specified time. Timed release status indicates that the job is being held in the queue for execution at a future time.

If no keyword is specified, /BY_JOB_STATUS displays the status of all jobs.

/DEVICE[=(keyword,...)]

Selects for display only those print jobs in the queue types specified. Specify the queue type with one or more of the following keywords:

PRINTER	Requests the display of jobs in print queues.
SERVER	Requests the display of jobs in server queues.
TERMINAL	Requests the display of jobs in terminal queues.

If no keyword is specified, /DEVICE displays all printer, terminal, and server queues. If /USER_NAME is not specified, information about your jobs is displayed.

/FILES

Adds to the default display the list of full file specifications for each file in each job.

/FULL

Displays the following information for each job: job name, user name, entry number, job status, full file specification associated with each job, date and time of submission, settings specified for the job, queue name, and queue type.

The /FULL qualifier overrides the default brief listing format.

/GENERIC

Selects for display only those jobs contained in generic queues. A generic queue holds jobs of a particular type (for example, batch or line printer jobs) and directs them to execution queues for processing. If /USER_NAME is not specified, information about your jobs is displayed.

/OUTPUT[=filespec]
/NOOUTPUT

Controls where the output of the SHOW ENTRY command is sent. By default, the output is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. The file specification may not include any wildcard characters. If you enter a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS is the default file type. If you enter /NOOUTPUT, output is suppressed.

/USER_NAME=username

Selects for display those jobs owned by the specified user. If /USER_NAME is not specified, information about your jobs is displayed.

example

```
$ SHOW ENTRY/DEVICE=(PRINTER,TERMINAL)
  Jobname         Username     Entry     Blocks    Status
  -------         --------     -----     ------    ------
  FORECAST        JONES          422         12    Printing
    On printer queue LNO1$PRINT

  MANAGER         JONES          431          4    Printing
    On terminal queue LQ$PRINT
```

In this example, SHOW ENTRY produces a display of your current job entries on all printer and terminal queues.

SHOW ERROR

Displays the error count for all devices with error counts greater than 0.

format

SHOW ERROR

parameters

None.

qualifiers

/FULL

Displays the error count for all devices, including those with no errors. (The error count is either 0 or a number greater than 0.)

/OUTPUT[=file-spec]
/OUTPUT=SYS$OUTPUT (default)

Specifies the file to which the display is written. By default, the display is written to the current SYS$OUTPUT device.

example

```
$ SHOW ERROR
```

Displays the error count for all devices with error counts greater than 0:

```
Device  Error Count
CPU         2
MEMORY      1
DBB1        9
```

SHOW INTRUSION

Displays the contents of the break-in database.

Requires the CMKRNL and SECURITY privileges.

format

SHOW INTRUSION

qualifiers

/OUTPUT[=file-spec]

Directs the output from the SHOW INTRUSION command to the file specified with the qualifier. By default, output from the command is displayed to SYS$OUTPUT.

/TYPE=keyword

Selects the type of information from the break-in database that is displayed. The valid keywords are as follows:

ALL	All break-in entries. By default, all entries are displayed.
SUSPECT	Break-in entries for login failures that have occurred but have not yet passed the threshold necessary to be identified as intruder.
INTRUDER	Break-in entries for which the login failure rate was high enough to warrant evasive action.

example

```
$  SHOW INTRUSION/TYPE=INTRUDER

Intrusion   Type      Count   Expiration    Source
TERMINAL    INTRUDER    9     10:29:39.16   _LTA23:
NETWORK     INTRUDER    7     10:47:53.12   STAR::HAMM
```

In this example, the SHOW INTRUSION command displays all intruder entries currently in the break-in database.

SHOW KEY

Displays the key definitions created with the DEFINE/KEY command.

format

SHOW KEY *[key-name]*

parameter

key-name

The name of the key whose definition you want displayed. See the DEFINE/KEY command for a list of valid key names.

qualifiers

/ALL

Displays all key definitions in the current state (or the state specified with the /STATE qualifier). If you use the /ALL qualifier, do not specify a key name.

/BRIEF (default)
/NOBRIEF

Displays only the key definition and state. The /BRIEF and /NOFULL qualifiers are equivalent.

/DIRECTORY

Displays the names of all states for which keys have been defined.

/FULL
/NOFULL (default)

Displays all qualifiers associated with a definition. By default, only the state of the definition and the definition itself are displayed. The /FULL and /NOBRIEF qualifiers are equivalent.

/STATE=(state-name[,...])
/NOSTATE

Displays the key definitions for the specified state. If you specify only one state name, you can omit the parentheses. State names can be any appropriate alphanumeric string. State names are created with the DEFINE/KEY command.

example

```
$ DEFINE/KEY/TERMINATE PF1 "ATTACH GEORGE"
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ SHOW KEY PF1
DEFAULT keypad definitions:
  PF1 = "ATTACH GEORGE"
$ SHOW KEY/FULL PF1
DEFAULT keypad definitions:
  PF1 = "ATTACH GEORGE"  (noecho,terminate,noerase,nolock)
```

The SHOW KEY command in this example displays both the definition and the state for the PF1 key. This is the default display. The SHOW KEY/FULL command displays all qualifiers associated with the key definition.

SHOW LOGICAL

Displays translations, the level of translation, and the logical name table for a specified logical name. The SHOW LOGICAL command performs iterative translations.

Requires READ (R) access to the table in which a logical name is cataloged to display information about the logical name.

format

SHOW LOGICAL *[logical-name[:][,...]]*

parameter

logical-name[:][,...]
Specifies one or more logical names whose translations you want to display. The asterisk (*) and percent (%) wildcard characters are allowed. However, if a wildcard character is used, iterative translation is not done.

qualifiers

/ACCESS_MODE=mode
Displays names defined in the specified access mode and any inner access modes. You can specify one of the following keywords to indicate the access mode: USER_MODE, SUPERVISOR_MODE, EXECUTIVE_MODE, or KERNEL_MODE.

/ALL (default)
Indicates that all logical names in the specified logical name tables are to be displayed.

/DESCENDANTS
/NODESCENDANTS (default)
Controls whether the system displays names from the specified logical name table and any descendant tables. A descendant table is created by the CREATE/NAME_TABLE command, with the /PARENT_TABLE qualifier specifying its parent table. If you use the /DESCENDANTS qualifier, you must also use the /TABLE qualifier.

/FULL
Displays more detailed information for the specified logical name. The information includes the access mode, attributes, the translation, and the logical name table.

/GROUP
Indicates that only the group logical name table is to be searched. The /GROUP qualifier is synonymous with /TABLE=LNM$GROUP. If you specify the /GROUP qualifier and you do not also specify a logical name, all names in the group table are displayed.

/JOB

Indicates that only the job logical name table is to be searched. The /JOB qualifier is synonymous with /TABLE=LNM$JOB. If you specify the /JOB qualifier and you do not also specify a logical name, all names in the job logical name table are displayed.

/OUTPUT[=file-spec]
/NOOUTPUT

By default, the output of the SHOW LOGICAL command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

/PROCESS

Indicates that only the process logical name table is to be searched. The /PROCESS qualifier is synonymous with /TABLE=LNM$PROCESS. If you specify the /PROCESS qualifier and you do not also specify a logical name, all names in the process table are displayed.

/STRUCTURE
/NOSTRUCTURE (default)

Controls whether the system displays the "family tree" of all accessible logical name tables. If you specify /STRUCTURE, you cannot use any other qualifiers except /ACCESS_MODE, /FULL, and /OUTPUT.

/SYSTEM

Indicates that only the system logical name table is to be searched. The /SYSTEM qualifier is synonymous with /TABLE=LNM$SYSTEM. If you specify the /SYSTEM qualifier and you do not also specify a logical name, all names in the system table are displayed.

/TABLE=(name[,...])

Specifies the tables you want to search. If you specify only one table, you can omit the parentheses. Wildcards are allowed. Names with wildcards are used to match table names. Names without wildcards are treated both as table names and table search lists (whichever is appropriate).

example

```
$ SHOW LOGICAL/PROCESS
(LNM$PROCESS_TABLE)
  "SYS$COMMAND" = "_TTB4:"
  "SYS$DISK" = "WORK6:"
  "SYS$DISK" = "WORK6:"
  "SYS$ERROR" = "_TTB4:"
  "SYS$INPUT" = "_TTB4:"
  "SYS$LOGIN" = "WORK6:[ODONNELL]"
  "SYS$LOGIN_DEVICE" = "WORK6:"
  "SYS$OUTPUT" = "_TTB4:"
  "SYS$OUTPUT" = "DBA2:"
  "SYS$SCRATCH" = "WORK6:[ODONNELL]"
```

The SHOW LOGICAL command in this example displays all process logical names and their translations. (Note that /TABLE=LNM$PROCESS would produce the same display as /PROCESS.)

SHOW MAGTAPE

Displays the current characteristics and status of a specified magnetic tape device.

format

SHOW MAGTAPE *device-name[:]*

parameter

device-name[:]
Specifies the name of the magnetic tape device for which you want to display the characteristics and status.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT
Specifies the file to which the display is written; by default, the display is written to the current SYS$OUTPUT device.

example

```
$ SHOW MAGTAPE MTAO:
  MTAO: UNKNOWN, DENSITY=800, FORMAT=Normal-11
        Odd Parity
```

The SHOW MAGTAPE command in this example displays the characteristics of the device MTA0:. The display shows the device type, density, and format (default or normal PDP-11).

It also displays the following characteristics:

Position lost	Write-locked
End-of-tape	Even parity
End-of-file	Odd parity
Beginning-of-tape	

SHOW MEMORY

Displays the availability and usage of those system resources that are related to memory.

format

SHOW MEMORY

qualifiers

/ALL (default)
Displays all available information, that is, information displayed by the /FILES, /PHYSICAL_PAGES, /POOL, and /SLOTS qualifiers.

/FILES
Displays information about the use of each paging and swap file currently installed.

/FULL
When used with the /POOL or /FILES qualifier, displays additional information about the use of each pool area or paging and swap file currently installed. This qualifier is ignored unless the /FILES or /POOL qualifier is explicitly specified.

/OUTPUT[=file-spec]
/NOOUTPUT
Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

/PHYSICAL_PAGES

Displays information about the amount of physical memory and the number of free and modified pages.

/POOL

Displays information about the usage of each dynamic memory (pool) area, including the amount of free space and the size of the largest contiguous block in each area.

/SLOTS

Displays information about the availability of PCB vector slots and balance slots.

example

```
$ SHOW MEMORY/SLOTS

            System Memory Resources on 31-DEC-1988 16:11:35.31
Slot Usage (slots):                Total❶      Free❷  Resident❸     Swapped❹
Process Entry Slots                  75          28        46          1
Balance Set Slots                    70          26        44          0
```

Slot Usage (slots)

Displays the use of process entry slots and balance slots.

❶Total — Displays the number of process entry slots (the value of the SYSGEN parameter MAXPROCESSCNT) and balance slots (the value of the SYSGEN parameter BALSETCNT) permanently allocated when the system was bootstrapped.

❷Free — Displays the number of slots currently not in use.

❸Resident — Displays the number of slots currently used by memory-resident processes. The number of balance slots in use can never be any larger than the number of process entry slots in use because the SWAPPER and NULL processes have process entry slots but do not require balance slots.

❹Swapped — Displays the number of slots used by outswapped processes. For process entry slots, this number includes all processes that have been partially outswapped. For balance slots, this number includes those processes that have had their process bodies outswapped but have process headers that are still resident.

SHOW NETWORK

Displays the availability of the local node as a member of the network [1] and the addresses and names of all nodes that are currently accessible to the local node. The SHOW NETWORK command also displays link and cost relationships between the local node and other nodes in the network.

format

SHOW NETWORK

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT. If you enter /OUTPUT with a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS is the default file type. If you enter a file specification, it may not include any wildcard characters. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW NETWORK

VAX/VMS Network Status for local node 2.161 ARAKIS on 15-APR-1988 09:18:03.07
The next hop to the nearest area router is node 2.62 ZEUS.
    Node          Links  Cost   Hops  Next Hop to Node
2.161  ARAKIS       0     0      0    Local    ->   2.161   ARAKIS
2.1    RAEL         0     8      1    UNA-0    ->   2.1     RAEL
2.2    PANGEA       0     8      1    UNA-0    ->   2.2     PANGEA
2.3    TWDEE        0    10      2    UNA-0    ->   2.63    AURORA
2.4    TWDUM        0     8      1    UNA-0    ->   2.4     TWDUM
2.11   NEONV        0     8      1    UNA-0    ->   2.11    NEONV
2.63   AURORA       0     8      1    UNA-0    ->   2.63    AURORA
            Total of 7 nodes.
```

If your local node is a nonrouting or end node and you enter the SHOW NETWORK command, the following message is displayed:

```
This is a nonrouting node, and does not have any network
information. The designated router for node _nodename is
node_number_name.
```

[1] DECnet–VAX is available under separate license.

SHOW PRINTER

Displays the current settings for a printer.

format

SHOW PRINTER *device-name[:]*

parameter

device-name[:]
Specifies the name of the printer for which settings are to be displayed.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT
By default, the output of the SHOW PRINTER command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW PRINTER LPA0:
 Printer LPA0:, device type LP11, is online, allocated, spooled
 Error count                    0  Operations completed              880
 Owner process  "SYMBIONT_0001"  Owner UIC                         [0,0]
 Owner process ID      21C0008D  Dev Prot  S:RWLP,O:RWLP,G:RWLP,W:RWLP
 Reference count               2  Default buffer size               132
 Page width                  132  Page Length                         66
 No Carriage_return     Formfeed        Lowercase
 No Passall             No Wrap         Printall
 No Fallback
 Intermediate device: STAR$DBA1:
 Associated queue: LN01$PRINT
```

The SHOW PRINTER command in this example displays the settings for the printer LPA0.

SHOW PROCESS

Displays information about a process and any current subprocesses. If no qualifier is entered, only a basic subset of information is displayed: the time, process terminal, user name and UIC, process name and process identification, priority, default directory, and allocated devices.

Requires GROUP privilege to show other processes in the same group. Requires WORLD privilege to show processes outside your group.

format

SHOW PROCESS *[process-name]*

parameter

process-name

Specifies the name of the process about which information is to be displayed. Process names can have from 1 to 15 alphanumeric characters. Process names are linked to group numbers. The specified process must have the same group number in its user identification code (UIC) as the current process.

qualifiers

/ACCOUNTING

Displays the accumulated accounting statistics for the current session.

/ALL

Displays the basic subset of information as well as accounting statistics, privileges, quotas, and subprocesses.

/CONTINUOUS

Displays continuously updated information about the process. While the continuous display is running, you can press the V key to display a map of the pages in the virtual address space of the process. To terminate the continuous display, press the E key. To return to the original display, press the space bar. The /CONTINUOUS qualifier may not be used with the /OUTPUT qualifier.

/IDENTIFICATION=pid

Requires GROUP or WORLD privilege to access processes other than your own. Displays information about the process with the specified PID (process identification). The PID is assigned by the system when the process is created. When you specify a PID, you can omit the leading zeros. If you specify the /IDENTIFICATION qualifier, you cannot use the process name parameter. If, in addition, you specify either the /MEMORY or /SUBPROCESSES qualifiers, the process identification (PID) value must be that of the current process.

/MEMORY

Displays the process's use of dynamic memory areas. The /MEMORY qualifier is allowed only for the current process.

/OUTPUT[=file-spec]
/NOOUTPUT

By default, the output of the SHOW PROCESS command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you

enter /NOOUTPUT, output is suppressed. The /OUTPUT qualifier may not be used with the /CONTINUOUS qualifier.

/PRIVILEGES

Displays current privileges for the process.

/QUOTAS

Displays, for each resource, either a quota or a limit. The values displayed for quotas reflect any quota reductions resulting from subprocess creation. The values displayed for limits reflect the resources available to a process at creation.

/SUBPROCESSES

Displays the current subprocesses in hierarchical order. This qualifier can be used only for the current process.

example

```
$ SHOW PROCESS
 21-FEB-1988 15:35:19.39   VTA102:              User: MALIK
 Pid: 28200364   Proc. name: MALIK             UIC: [VMS,MALIK]
 Priority:   4   Default file spec: WORK5:[MALIK]
 Devices allocated: VTA102:
```

The SHOW PROCESS command in this example displays the basic subset of information for the current process.

SHOW PROTECTION

Displays the current file protection to be applied to all new files created during the terminal session or batch job. You can change the default protection at any time with the SET PROTECTION command.

format

SHOW PROTECTION

example

```
$ SHOW PROTECTION
  SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$ SET PROTECTION=(GROUP:RWED,WORLD:RE)/DEFAULT
$ SHOW PROTECTION
  SYSTEM=RWED, OWNER=RWED, GROUP=RWED, WORLD=RE
```

The SHOW PROTECTION command in this example requests a display of the current protection defaults and the user identifiers; the SET PROTECTION/DEFAULT command changes the file access allowed to other users in the same group and to miscellaneous system users. The next SHOW PROTECTION command shows the modified protection defaults.

SHOW QUEUE

Displays information about queues and the jobs that are currently in queues.

Requires GROUP privilege to show all jobs in your group. Requires OPER privilege to show all jobs in all groups.

format

SHOW QUEUE *[queue-name]*

parameter

queue-name

Specifies the name of the queue for which you want information displayed. Wildcard characters (* and %) are allowed. The default value for the queue-name parameter is the asterisk wildcard (*). If no queue name is specified, information on all queues is displayed.

qualifiers

/ALL_JOBS

Displays all the jobs in the specified queues. If you do not specify a queue name, the /ALL_JOBS qualifier displays all job entries on all queues. To modify the display, combine this qualifier with the /BY_JOB_STATUS qualifier.

/BATCH

Displays all batch queues. Use the /BATCH qualifier in conjunction with other qualifiers to display specific information about particular batch queues.

/BRIEF

Displays a one line description of each queue and the jobs that are in it. This information includes the name, type, and status of each queue. It also includes the user name, entry number, and status for each job. The /FULL and /FILES qualifiers override /BRIEF.

/BY_JOB_STATUS=(keyword-list)

Displays jobs with the specified status. Specify the status with one or more of the following keywords:

EXECUTING	Requests the display of currently executing jobs.
HOLDING	Requests the display of jobs on hold. Holding status indicates that the job is being held in the queue indefinitely.
PENDING	Requests the display of jobs with pending status. Pending status indicates that the job is waiting its turn to execute.
RETAINED	Requests the display of jobs retained in the queue after execution. Retained status indicates that the job has completed, but it remains in the queue. For example, a job may be retained in the queue if there was an error during its execution.
TIMED_RELEASE	Requests the display of jobs on hold until a specified time. Timed release status indicates that the job is being held in the queue for execution at a future time.

Note that if you specify the qualifier as /BY, the system will only display queues that actually contain jobs.

/DEVICE[=(keyword-list)]

Displays a particular type of queue. Use the /DEVICE qualifier in conjunction with other qualifiers to display specific information about particular device queues.

Specify the type of device queue with one or more of the following keywords:

PRINTER	Requests the display of all print queues.
SERVER	Requests the display of all server queues.
TERMINAL	Requests the display of all terminal queues.

You can specify more than one keyword. If you do not specify a keyword, /DEVICE displays all printer, terminal, and server queues.

/FILES

Adds to the display the list of files associated with each job.

/FULL

Displays complete information about queues, the jobs contained in queues, and the files associated with the jobs. See /BRIEF.

/GENERIC

Displays all generic queues. A generic queue is not an execution queue. Its function is to hold jobs of a particular type (line printer jobs, for example) and direct them to execution queues for processing.

Use the /GENERIC qualifier in conjunction with other qualifiers to display specific information about particular generic queues. For example, use the /GENERIC qualifier along with the /BATCH qualifier to specify information about generic batch queues. Use the /GENERIC qualifier along with the /DEVICE qualifier to determine information concerning generic output queues.

/OUTPUT[=file-spec]
/NOOUTPUT

By default, the output of the SHOW QUEUE command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

/SUMMARY

Displays the total number of executing jobs, pending jobs, holding jobs, retained jobs, and timed release jobs for each queue. For output queues, the total block count for pending jobs is also shown.

example

```
$ SHOW QUEUE/FULL CAXTON_LPA0

Printer queue CAXTON_LPA0, on CAXTON::CAXTON_LPA0, mounted form
80_COLS (stock=BLUE)
    /BASE_PRIORITY=100
    /DEFAULT=(FEED,FLAG,FORM=40_COLS (stock=WHITE),TRAILER=ONE)
    /NOENABLE_GENERIC Lowercase /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  ACCOUNT         MARTIN         880      10  Printing
    Submitted  9-AUG-1988 12:49 /FORM=80_COLS (stock=BLUE)  /PRIORITY=100
  REPORT          MARTIN         858       4  Pending
    Submitted  8-AUG-1988 17:27 /PRIORITY=100
```

The SHOW QUEUE command in this example lists any current job entry you have on the printer queue CAXTON_LPA0. The /FULL qualifier lists the submission information, the full file specification, and the current settings for both the job and the queue.

SHOW QUEUE/CHARACTERISTIC

Displays information about queue characteristics defined for the system. A characteristic is a user-defined attribute of a batch or output queue, such as ink color.

format

SHOW QUEUE/CHARACTERISTIC
[characteristic-name]

parameter

characteristic-name

Specifies the name of a characteristic. Wildcard characters (* and %) are allowed. The default value for the characteristic-name parameter is the asterisk wildcard (*). Thus, information about all characteristics is displayed when you do not specify a characteristic name.

qualifier

/OUTPUT[=filespec]
/NOOUTPUT
By default the output of the SHOW QUEUE/CHARACTERISTIC command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW QUEUE/CHARACTERISTIC *INK
Characteristic name                 Number
-------------------                 ------
REDINK                                   0
BLUEINK                                  6
BROWNINK                                25
```

The SHOW QUEUE/CHARACTERISTIC command in this example displays the name and number of all characteristics that end with INK.

SHOW QUEUE/FORM

Displays information about forms defined for the system. Forms define the size and type paper and the layout of text that are used for print jobs.

format

SHOW QUEUE/FORM *[form-name]*

parameter

form-name
Specifies the name of the form. Wildcard characters are allowed. The default value for the form-name parameter is an asterisk (*) which means that the names of all forms on the system are displayed.

qualifiers

/BRIEF (default)
Displays a brief description (form names, numbers, and descriptions) about the forms on the system.

/FULL
Displays a full description (including paper size and margin settings) about the forms on the system.

/OUTPUT[=file-spec]
/NOOUTPUT
By default the output of the SHOW QUEUE/FORM command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output

to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW QUEUE/FORM/FULL
Form name                                Number   Description
---------                                ------   -----------
132_51_STD (stock=DEFAULT)                 102    132 by 51 (standard short)
    /LENGTH=51 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132
40_66_STD (stock=DEFAULT)                  103    40 by 66 (standard labels)
    /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /WIDTH=40
BLUE_PAPER_STOCK (stock=DIGITAL_8X11_STOCK1412TEA)
                                         22222    blue paper, DEC order# 22222
    /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DIGITAL_8X11_STOCK1412TEA
    /TRUNCATE  /WIDTH=80
DEFAULT                                    0    System-defined default
    /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132
LNO1_LANDSCAPE (stock=DEFAULT)             105    132 by 66 (landscape)
    /LENGTH=66 /STOCK=DEFAULT /WIDTH=132
LNO1_LANDSCAPE_INDENTED (stock=DEFAULT)
                                           107    132 by 65 (landscape)
    /LENGTH=65 /SETUP=(LNO1_TOP_MARGIN_150) /STOCK=DEFAULT /WIDTH=132
LNO1_PORTRAIT (stock=DEFAULT)              106    80 by 60 (portrait)
    /LENGTH=60 /SETUP=(LNO1_PORTRAIT) /STOCK=DEFAULT /WIDTH=80
MEMO (stock=DEFAULT)                       110    LNO3 indented memo format
    /LENGTH=64 /MARGIN=(TOP=2,LEFT=5) /STOCK=DEFAULT /TRUNCATE /WIDTH=80
```

This SHOW QUEUE/FORM command also displays the names of all form types and stock for the system. By using the /FULL qualifier, you can see what image size has been set for each form type.

SHOW QUOTA

Displays the current disk quota that is authorized for a specific user on a specific disk. This display includes a calculation of the amount of space available and the amount of overdraft that is permitted.

Requires READ (R) access to the quota file in order to display the quotas of other users.

format

SHOW QUOTA

qualifiers

/DISK[=device-name[:]]

Specifies the disk whose quotas are to be examined. By default, the current default disk (defined by SYS$DISK) is examined.

/USER=uic

Specifies which user's quotas are to be displayed. By default, the current user's quotas are displayed.

example

```
$ SHOW QUOTA /USER=[360,007]/DISK=XXX1:
%SYSTEM-F-NODISKQUOTA, no disk quota entry for this UIC
```

The SHOW QUOTA command in this example displays the fact that the user with UIC [360,007] has no disk quota allocation on device XXX1.

SHOW RMS_DEFAULT

Displays the current default values for the multiblock count, the multibuffer count, the network transfer size, the prolog level, and the extend quantity.

format

SHOW RMS_DEFAULT

parameters

None.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Specifies the file to which the display is written (default is SYS$OUTPUT). Wildcard characters are not allowed.

example

```
$ SHOW RMS_DEFAULT

          MULTI-  |                MULTIBUFFER COUNTS                | NETWORK
          BLOCK   | Indexed  Relative          Sequential           |  BLOCK
          COUNT   |                      Disk   Magtape  Unit Record |  COUNT
Process     0     |    0        0         0       0         0       |    0
System     16     |    0        0         0       0         0       |    8

           Prolog     Extend Quantity
Process      0              0
System       0              0
```

The SHOW RMS_DEFAULT command in this example shows a system multiblock count of 16 and a network block count of 8. These are typical values.

SHOW STATUS

The SHOW STATUS command in this example displays the current status of your process.

format

SHOW STATUS

parameters

None.

example

```
$ SHOW STATUS
  Status on  15-APR-1988 12:56:48.68        Elapsed CPU :   0 00:00:55.02
  Buff. I/O :      5117    Cur. ws. :     300        Open files :        1
  Dir. I/O :        458    Phys. Mem. :   162        Page Faults :    8323
```

Displays the status of your process. The information includes the following:

- Current time and date
- Elapsed CPU time used by the current process
- Buffered I/O count
- Current working set size
- Open file count
- Direct I/O count
- Current amount of physical memory occupied
- Number of page faults

SHOW SYMBOL

Displays the value of the specified symbol.

format

SHOW SYMBOL *[symbol-name]*

parameter

symbol-name

Specifies the name of the symbol whose value you want to display. You must specify a symbol name unless you use the /ALL qualifier. Wildcard characters are allowed in the symbol-name parameter.

qualifiers

/ALL
Displays the current values of all symbols in the specified symbol table (/LOCAL or /GLOBAL). If you specify /ALL and do not specify either /LOCAL or /GLOBAL, the SHOW SYMBOL command displays the contents of the local symbol table for the current command level.

/GLOBAL
Searches only the global symbol table for the specified symbol name. If you specify both the /ALL and /GLOBAL qualifiers, all names in the global symbol table are displayed.

/LOCAL
Searches only the local symbol table for the current command level for the specified symbol name. If you specify both the /ALL and /LOCAL qualifiers, all names in the local symbol table for the current command level are displayed.

/LOG (default)
/NOLOG
Controls whether the system generates an informational message if the symbol value has been truncated. The value is truncated if it exceeds 255 characters.

example

```
$ SHOW SYMBOL/GLOBAL/ALL
  TIME == "SHOW TIME"
  LOG == "@LOG"
  $RESTART == "FALSE"
  $SEVERITY == "1"
  $STATUS == "%X00000001"
```

The SHOW SYMBOL command in this example displays all the symbols defined in the global symbol table. Note that the symbols $RESTART, $STATUS, and $SEVERITY, which are maintained by the system, are also displayed.

SHOW SYSTEM

Displays status information about current processes: the time, process name and identification, processing state, priority, total process I/O, cumulative processor time used, cumulative page faults, amount of physical memory being used, and type of process.

format

SHOW SYSTEM

parameters

None.

qualifiers

/BATCH
Displays all batch jobs in the system.

/FULL
Displays the user identification code (UIC) in addition to the default information. The UIC is displayed underneath the process name.

/NETWORK
Displays all network processes in the system.

/OUTPUT[=file-spec]
/NOOUTPUT
By default, the output of the SHOW SYSTEM command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

/PROCESS (default)
Displays all processes in the system.

/SUBPROCESS
Displays all subprocesses in the system.

example

```
$ SHOW SYSTEM
VMS V5.0  on node CAXTON 31-DEC-1988 15:10:31.02   Uptime     0 12:06:30
  Pid     Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
22200081 SWAPPER          HIB     16        0    0 00:01:07.92         0     0
22200202 Meg              LEF      4     6206    0 00:03:52.53     17509   174
22200085 ERRFMT           HIB      8     4123    0 00:00:45.35       145   191
22200086 CLUSTER_SERVER   HIB     10       23    0 00:00:01.63       130   167
22200087 OPCOM            RWAST    9     1114    0 00:00:39.60       189   274
22200088 JOB_CONTROL      HIB      8     6662    0 00:02:29.09       368   474
22200089 CONFIGURE        HIB     13       84    0 00:00:00.64       148   235
2220008A VAXsim_Monitor   HIB      8     1543    0 00:00:20.55       312   137
2220008C PARTITION_CHKR   LEF      4     3833    0 00:00:14.27       132   153
2220008D SYMBIONT_0001    HIB      4     2866    0 00:16:26.65      6751   329
2220008E Cerb Servant     LEF      4    12412    0 00:03:08.40       854   165
2220008F Monitor          LEF     15     1151    0 00:00:28.64      2396   350
22200090 NETACP           HIB      9    14820    0 00:02:42.85      8464   350
22200091 EVL              HIB      5      209    0 00:00:07.90      9996    49   N
22200092 REMACP           HIB      9      194    0 00:00:01.18       123    59
22200113 BIGELOW          PFW      4    16505    0 00:04:17.12     23318   912
22200119 OHARE            LEF      9     7680    0 00:01:36.09      8689   233
2220011D BRATZ            LEF      9     6276    0 00:00:43.66      2006   200
2220019E PENN             LEF      4     2646    0 00:01:59.11      4847   280
222001A4 TAMMY            LEF      9    13514    0 00:01:49.52      3011   512
222001A6 C_EMACS          HIB      7    16165    0 00:05:10.86      6026   639   S
222001A7 PERKINS          LEF      4     7572    0 00:02:13.84     16487   195
22200128 SAPP             LEF      4    15634    0 00:07:34.16     39591   450
2220022C ZIPPY            LEF      7     2297    0 00:00:52.86      6668   300
2220012E GENIUS           LEF      9      671    0 00:00:12.02      1197   200
2220022F BORSE            LEF      4      741    0 00:00:13.13      2394   150
22200231 MSCPmount        LEF      6     1211    0 00:00:48.40      3793    92
222000B2 MCGOY            LEF      4    11710    0 00:01:54.90     12798   180
222000B5 pf cobbs ttg2    LEF      4     6076    0 00:01:26.10      8418   220
22200236 COBBS            HIB      8    39982    0 00:00:31.73      5419   452
22200236 COBBS_1          CUR 1    4      251    0 00:00:05.80       560   330
22200237 MAIL_6188        LEF      6      236    0 00:00:04.58       609   269   N
22200238 MAIL_6189        LEF      4      236    0 00:00:04.58       542   261   N
22200239 Mike Smith TTG   COM      4      220    0 00:00:05.42       844   154
222000BC Greg A-z         LEF      9     9873    0 01:35:16.42     16557   949
222001C5 BUNNYRABBIT      LEF      9      492    0 00:00:12.00      2171   150
222000C8 CARP             LEF      8     9381    0 00:05:44.12     22090   227
2220014A CATFISH          LEF      9     9512    0 00:02:03.49     10264   227
222000CB TUNA             LEF      9     5676    0 00:01:56.31      9468   526
222000CC HANGNAIL         LEF      9    43511    0 00:16:30.55     55396   163
222000D2 BAZOO            LEF      4    17315    0 00:05:25.43     62535   300
222000DE KENNY            LEF      7     4887    0 00:01:39.36      8800   544
2220015F MACY             CUR 0    4    13671    0 00:03:10.51     19027   302
22200162 _TTH1:           LEF      4    19344    0 00:20:57.28     54894   269
222000E4 BATCH_970        LEF      6      669    0 00:00:09.79       639   386   B
22200167 EPPY             HIB      5     3399    0 00:01:01.07     10761   192
222000E9 Greg A_z         LEF      6    12899    0 00:01:52.69      6624   447
```

The SHOW SYSTEM command in this example displays all processes on the system.

The information in this example includes the following:

- Process identification code (PID)—a 32-bit binary value that uniquely identifies a process.
- Process name—a 1 to 15 character string used to identify a process.
- Process state—the activity level of the process, such as COM (computing), HIB (hibernation), LEF (local event flag) wait, or CUR (if the process is current). If a multiprocessing environment exists, the display shows the CPU ID of the processor on which any current process is executing.
- Current priority—the priority level assigned to the process (the higher the number, the higher the priority). [1]
- Total process I/O count [1]—the number of I/O operations involved in executing the process. This consists of both the direct I/O count and the buffered I/O count.
- Charged CPU time [1]—the amount of CPU time that a process has used thus far.
- Number of page faults [1]—the number of exceptions generated by references to pages which are not in the process's working set.
- Physical memory occupied [1]—the amount of space in physical memory that the process is currently occupying.
- Process indicator—letter B indicates a batch job; letter S indicates a subprocess; letter N indicates a network process.
- User identification code (UIC)—an 8-digit octal number assigned to a process. This is only displayed if the /FULL qualifier is specified.

SHOW TERMINAL

Displays the current characteristics of a specific terminal. Each characteristic corresponds to an option of the SET TERMINAL command.

format

SHOW TERMINAL *[device-name[:]]*

[1] This information is displayed only if the process is currently in the balance set; if the process is not in the balance set, these columns contain the following message:

– swapped out –

parameter

device-name[:]

Specifies the name of the terminal for which you want the characteristics displayed. The default is your terminal (SYS$COMMAND).

qualifiers

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT. If you enter /OUTPUT with a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS the default file type. If you enter a file specification, it may not include any wildcard characters. If you enter /NOOUTPUT, output is suppressed.

/PERMANENT

Requires LOG_IO or PHY_IO privilege. Displays the permanent characteristics of the terminal.

example

```
$ SHOW TERMINAL
Terminal: _TTE4:        Device_Type: VT102          Owner: FRANKLIN
Physical Terminal: _LTA49
   Input:   9600      LFfill:  0      Width:  80      Parity: None
   Output:  9600      CRfill:  0      Page:   24
Terminal Characteristics:
   Interactive        Echo                Type_ahead          No Escape
   No Hostsync        TTsync              Lowercase           Tab
   Wrap               Scope               No Remote           Eightbit
   Broadcast          No Readsync         No Form             Fulldup
   No Modem           No Local_echo       No Autobaud         Hangup
   No Brdcstmbx       No DMA              No Altypeahd        Set_speed
   Line Editing       Overstrike editing  No Fallback         No Dialup
   No Secure server   No Disconnect       No Pasthru          No Syspassword
   No SIXEL Graphics  Soft Characters     Printer port        Numeric Keypad
   ANSI_CRT           No Regis            No Block_mode       Advanced_video
   Edit_mode          DEC_CRT             DEC_CRT2            No DEC_CRT3
```

In this example, the SHOW TERMINAL command displays the characteristics of this specific terminal. If you are displaying statistics about a terminal allocated to another user, the input, output, LFfill, CRfill, width, page, and parity statistics are not shown.

SHOW TIME

Displays the current date and time. The DAY element is optional.

format

SHOW [DAY]TIME

parameters

None.

example

```
$ SHOW TIME
  4-FEB-1988 00:03:45
```

The SHOW TIME command in this example displays the current date and time.

SHOW TRANSLATION

Displays the first translation found for the specified logical name. You can specify the tables that are searched.

Requires READ (R) access to a logical name table to display information about any logical name cataloged in that table.

format

SHOW TRANSLATION *logical-name*

parameter

logical-name
Specifies the logical name whose translation you want to display.

qualifier

/TABLE=name
Searches the specified table. The default is /TABLE=LNM$DCL_LOGICAL.

example

```
$ SHOW TRANSLATION/TABLE=LNM$SYSTEM USER
  USER = "DBA2:"  (LNM$SYSTEM_TABLE)
```

The SHOW TRANSLATION command in this example displays the translation for the logical name USER. Because a table name is specified, the SHOW TRANSLATION command does not use the default search order. Only the specified table, LNM$SYSTEM, is searched. LNM$SYSTEM is the system logical name table.

SHOW USERS

Displays the terminal name, user name, and process identification code (PID) of interactive users on the system.

format

SHOW USERS *[username]*

parameter

username
Specifies the user about whom you want information. If you specify a string, all users whose user names begin with the string are displayed. If you omit the username parameter, a list of all interactive users is displayed.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT
By default, the output of the SHOW USERS command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW USERS GOSHGARIAN
          VAX/VMS Interactive Users
           31-DEC-1988 16:45:14.14
    Total number of interactive users = 32

   PID     Username      Process Name    Terminal
 20200115 GOSHGARIAN     GOSHGARIAN       VTA3:          TTA7:
```

The SHOW USERS command in this example displays the process identification code (PID), the user name, process name, and terminal names of the interactive user GOSHGARIAN.

SHOW WORKING_SET

Displays the working set limit, quota, and extent assigned to the current process.

format

SHOW WORKING_SET

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

example

```
$ SHOW WORKING_SET
  Working Set      /Limit= 180   /Quota= 350            /Extent= 1200
  Adjustment enabled    Authorized Quota= 350  Authorized Extent= 1200
```

In this example, the response to the SHOW WORKING_SET command indicates that the current process has a working set limit of 180 pages, a quota of 350 pages and that the current quota is equal to the authorized limit (350 pages). It also shows that the current process has a working set extent of 1200 and that the current extent is equal to the authorized limit (1200).

SORT

Invokes the Sort/Merge Utility (SORT) to reorder the records in a file into a defined sequence and to create either a new file of the reordered records or an address file by which the reordered records can be accessed. For a complete description of the Sort/Merge Utility, including more information about the SORT command, see the Reference Section.

format

SORT *input-file-spec[,...] output-file-spec*

SPAWN

Creates a subprocess of the current process.

The RESOURCE_WAIT state is required to spawn a process. Requires TMPMBX or PRMMBX user privilege. The SPAWN command does not manage terminal characteristics. The SPAWN and ATTACH commands cannot be used if your terminal has an associated mailbox.

format

SPAWN *[command-string]*

parameter

command-string
Specifies a command string of less than 132 characters that is to be executed in the context of the created subprocess. When the command completes execution, the subprocess terminates and control returns to the parent process. If both a command string and the /INPUT qualifier are specified, the specified command string executes before additional commands are obtained from the /INPUT qualifier.

qualifiers

/CARRIAGE_CONTROL
/NOCARRIAGE_CONTROL
Determines whether carriage return/line feed characters are prefixed to the subprocess's prompt string. By default, SPAWN copies the current setting of the parent process.

/CLI=cli-file-spec
/NOCLI
Specifies the name of a command language interpreter (CLI) to be used by the subprocess. The default CLI is the same as the parent process (defined in SYSUAF). If you specify /CLI, the attributes of the parent process are copied to the subprocess.

/INPUT=file-spec
Specifies an input file containing one or more DCL commands to be executed by the spawned subprocess. File type defaults to COM and no wildcards are allowed in the file specification. Once processing of the input file is complete, the subprocess is terminated. If both a command string and the /INPUT qualifier are specified, the specified command string executes before additional commands are obtained from the /INPUT qualifier. If neither is specified, SYS$INPUT is assumed (in which case a SPAWN/NOWAIT command is aborted if CTRL/Y is pressed to abort something running in your parent process).

/KEYPAD (default)
/NOKEYPAD
Copies keypad key definitions and the current keypad state from the parent process.

/LOG (default)
/NOLOG
Displays the assigned subprocess name and any messages indicating transfer of control between processes.

/LOGICAL_NAMES (default)
/NOLOGICAL_NAMES

Copies process logical names and logical name tables to the subprocess. By default, all process logical names and logical name tables are copied to the subprocess except those explicitly marked CONFINE or created in executive or kernel mode.

/NOTIFY
/NONOTIFY (default)

Controls whether a message is broadcast to your terminal notifying you that your subprocess has completed or aborted. This qualifier should not be used unless you specify the /NOWAIT qualifier. /NOTIFY cannot be specified when the SPAWN command is executed from within a noninteractive process.

/OUTPUT=file-spec

Specifies the output file to which the results of the SPAWN operation are written. No wildcards can be used in the file specification. (Do not specify SYS$COMMAND as a file specification for /OUTPUT when using the /NOWAIT qualifier; both parent and subprocess output will be displayed simultaneously on your terminal.)

/PROCESS=subprocess-name

Specifies the name of the subprocess to be created. The default subprocess name format is username_n.

/PROMPT[=string]

Specifies the prompt string for DCL to use in the subprocess. The default is the prompt of the parent process. The string must be enclosed in quotation marks if it contains spaces, special characters, or lowercase characters.

/SYMBOLS (default)
/NOSYMBOLS

Determines whether global and local symbols (except $RESTART, $SEVERITY, and $STATUS) are passed to the subprocess.

/TABLE=command-table

Specifies the name of an alternate command table to be used by the subprocess.

/WAIT (default)
/NOWAIT

Requires that you wait for the subprocess to terminate before you enter another DCL command. The /NOWAIT qualifier allows you to enter new commands while the subprocess is running. (Use the /OUTPUT qualifier with the /NOWAIT qualifier to avoid displaying both parent and subprocess output on the terminal simultaneously.)

example

```
$ RUN MYPROG
    .
    .
    .
$ CTRL/Y
$ SPAWN MAIL
%DCL-S-SPAWNED, process SMITH_1 spawned
%DCL-S-ATTACHED, terminal now attached to process SMITH_1
MAIL> READ
    .
    .
    .
MAIL> EXIT
%DCL-S-RETURNED, control returned to process SMITH
$ CONTINUE
```

The SPAWN command in this example allows you to enter the VMS Mail Utility without terminating the currently running program. After you exit from MAIL, control is returned to the parent process.

START/CPU

Starts the specified secondary processor or processors in a VMS multiprocessing system. The /CPU qualifier is required.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

START/CPU *[cpu-id,...]*

parameter

[cpu-id,...]
Decimal value representing the identity of a processor in a VMS multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0. If you do not specify a CPU ID and do not include the /ALL qualifier, the START/CPU command selects a single available processor to join the multiprocessing system.

description

The START/CPU command starts a secondary processor in a VMS multiprocessing system.

You can issue a START/CPU command only for processors in the STOPPED or TIMOUT state, as represented by the SHOW CPU command. Otherwise, the START/CPU command has no effect.

qualifier

/ALL

Selects all remaining processors in the system's available set to join the multiprocessing system.

START/QUEUE

Starts or restarts the specified queue after it has been initialized. The /QUEUE qualifier is required.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

START/QUEUE *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue to be started or restarted.

qualifiers

/ALIGN[=(option[,...])]

Prints alignment pages that enable the operator to align properly the forms in the printer or terminal. Use this qualifier to restart an output queue from a paused state. Possible options are as follows:

MASK	Displays alphabetic characters as x's and numbers as 9's; nonalphanumeric characters are not masked. The default is not to mask.
n	Specifies the number of alignment pages to print. The value of *n* can be from 1 to 20; the default is 1.

/BACKWARD=n

Restarts a print queue *n* pages before the current page; *n* defaults to 1. Use this qualifier in restarting an output queue from a paused state.

/BASE_PRIORITY=n

Specifies the base process priority at which jobs are initiated from a batch queue or the base priority of the symbiont process for printer, terminal, or server queues. By default, if you omit the qualifier, jobs are initiated at the same priority as the base priority established by DEFPRI at system generation (usually 4). The value n can be any decimal value from 0 through 15.

/BATCH
/NOBATCH (default)

Indicates that this is a batch queue. The /NOBATCH qualifier cancels the effect of a previous /BATCH qualifier on the same command. It is supported in this release for compatibility with VMS Version 4.n. The function of the /[NO]BATCH qualifier has been incorporated into the /[NO]BATCH qualifier of the INITIALIZE/QUEUE command. DIGITAL recommends that you use this command to determine queue type and that existing command procedures using START/QUEUE/[NO]BATCH be updated.

/BLOCK_LIMIT=([lowlim,]uplim)
/NOBLOCK_LIMIT

Restricts the size of print jobs that can be executed on a printer or terminal queue. You must specify at least one of the parameters. The lower parameter is a decimal number referring to the minimum number of blocks that are accepted by the queue for a print job. The upper parameter is a decimal number referring to the maximum number of blocks that are accepted by the queue for a print job. If a job contains fewer blocks than the number specified by the lower parameter or more blocks than the number specified by the upper parameter, the job remains pending until the block limit for the queue is changed, enabling the job to execute.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS

Specifies one or more characteristics for processing jobs on the queue. Each time you specify /CHARACTERISTIC, all previously set characteristics are erased. A queue must have all the characteristics specified for the job or the job remains pending. Only the characteristics specified with the qualifier are now established for the queue. If only one characteristic is specified, you can omit the parentheses. Queue characteristics are installation-specific. The characteristic parameter can be either a value from 0 through 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.

/CLOSE

Prevents jobs from being entered in the queue through PRINT or SUBMIT commands or as a result of requeue operations. To allow jobs to be entered, use the /OPEN qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled). When a queue is marked closed, jobs executing continue to execute. Jobs already pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Indicates the default CPU time limit for batch jobs. Time can be specified as delta time, 0, NONE, or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time (subject to the restrictions imposed by the /CPUMAXIMUM qualifier or the user authorization file); the keyword NONE indicates that no time limit is needed. The value for time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier.

/CPUMAXIMUM=time

Indicates the maximum CPU time limit for batch jobs. The /CPUMAXIMUM qualifier overrides the time limit specified in the user authorization file (UAF). Time can be specified as delta time, 0, NONE, or INFINITE. You can specify up to 497 days of delta time. Both the value 0 and the keyword INFINITE allow unlimited CPU time; the keyword NONE specifies that no time limit is needed.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. Once an option is set for the queue by the /DEFAULT qualifier, users do not have to specify that option in their PRINT commands. The /DEFAULT qualifier cannot be used with the /GENERIC qualifier. Possible options are as follows:

[NO]BURST[=keyword]	Specifies whether to print burst pages (flag pages printed over the paper's perforations for easy identification of individual files in a print job). The keyword ALL (the default) places burst pages before each printed file in the job. The keyword ONE places a burst page before the first printed file in the job.
[NO]FEED	Specifies whether a form-feed is automatically inserted at the end of a page
[NO]FLAG[=keyword]	Specifies whether to print flag pages (containing the job entry number, the name of the user submitting the job, and so on). The keyword ALL places flag pages before each printed file in the job. The keyword ONE places a flag page before the first printed file in the job.
FORM=type	Specifies the default form for a printer, terminal, or server queue. If a job is not submitted with an explicit form definition, then this form is used to process the job. The systemwide default form, form=0, is the default value for this keyword. See also /FORM_MOUNTED.
[NO]TRAILER[=keyword]	Specifies whether to print trailer pages. The keyword ALL places trailer pages after each printed file in the job. The keyword ONE places a trailer page after the last printed file in the job.

/DESCRIPTION=string
/NODESCRIPTION (default)
A string of up to 255 characters used to provide operator-supplied information about the queue.

If the string contains alphanumeric, underscore, or dollar sign characters it must be enclosed in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may have been associated with the queue.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING (default)
Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC (default)
/NOENABLE_GENERIC
Allows files queued to a generic queue that does not specify explicit queue names in the /GENERIC qualifier to be placed in this execution queue for processing.

/FORM_MOUNTED=type
Specifies the form type for a printer, terminal, or server queue. If the stock of the mounted form is not identical to the stock of the default form, as indicated by the DCL command qualifier /DEFAULT=FORM=type, then all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, then the job enters a pending state. In both cases, the pending state is maintained until the stock of the mounted form of the queue is identical to the stock of the form associated with the job. Specify the form type using a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific. The /FORM_MOUNTED qualifier cannot be used with the /GENERIC qualifier.

/FORWARD=n
Advances the specified number of pages before resuming printing the current file in the current job; the default is 1. Use this qualifier to restart an output queue from a paused state.

/GENERIC[=(queue-name[,...])]
/NOGENERIC (default)
Specifies that this is a generic queue and that jobs placed in it can be moved for processing to compatible execution queues. The /GENERIC qualifier optionally accepts a list of target execution queues that have been previously defined. For a generic batch queue, these target queues must be batch execution queues. For a generic output queue, these target queues must be output execution queues, but can be of any type (printer, server, or terminal). If you do not specify any target queues with the /GENERIC

qualifier, jobs can be moved to any execution queue that (1) is initialized with the /ENABLE_GENERIC qualifier, and (2) is the same type (batch, printer, server, or terminal) as the generic queue. Moreover, for a generic server queue, an additional check is made: the symbiont named with the /PROCESSOR qualifier must be the same for both the generic and execution queues. The /GENERIC qualifier is used in conjunction with either the /BATCH or /DEVICE qualifiers to define the queue as a generic batch, printer, server, or terminal queue. If neither /BATCH nor /DEVICE is specified on creation of a generic queue, it becomes a generic printer queue by default.

/JOB_LIMIT=n

Specifies the number of batch jobs that can be executed concurrently from the queue. The job limit default value for n is 1.

/LIBRARY=file-name
/NOLIBRARY

Specifies the file name for the device control library. When you are initializing a symbiont queue, you can use the /LIBRARY qualifier to specify an alternate device control library. The default library is SYS$LIBRARY:SYSDEVCTL.TLB. You can specify only a file name as the parameter of the /LIBRARY qualifier. The system always assumes that the location of the file is in SYS$LIBRARY and that the file type is TLB.

/NEXT

Restarts the queue with the next job. By default, the job that was executing when the queue stopped resumes printing if it has not been deleted.

/ON=[node::]device[:] (printer, terminal, server queue)
/ON=node:: (batch queue)

Specifies the node or device, or both, on which this execution queue is located. For batch queues, only the node name can be specified. You can include both the node name and the device name for printer and terminal queues. By default, a queue executes on the same node from which you first start the queue. The default device parameter is the same as the queue name.

/OPEN (default)

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, stalled).

/OWNER_UIC=uic

Requires OPER privilege. Enables you to change the UIC of the queue. The default UIC is [1,4].

/PROCESSOR=file-name
/NOPROCESSOR

Allows users to specify their own print symbionts. The file name specifier can be any valid file name. The system supplies the device and directory name SYS$SYSTEM as well as the file type EXE. If you use this qualifier for an output queue, it specifies that the symbiont image to be executed is SYS$SYSTEM:file-name.EXE. By default, SYS$SYSTEM:PRTSMB.EXE is executed. If you use this qualifier for a generic queue, it specifies that the generic queue can place jobs only on queues established as server queues and that are executing the specified symbiont image.

/PROTECTION=(codes)

Requires OPER privilege. Specifies the protection of the queue. By default, the queue protection is (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W).

/RECORD_BLOCKING (default)
/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify /NORECORD_BLOCKING, the symbiont is directed to send each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN (default)

Retains jobs in the queue in a completed status after they have executed. Possible options are as follows:

ALL	Retains all jobs in the queue after execution (default)
ERROR	Retains in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in a printer or terminal queue are scheduled for printing based on the size of the job. When the default, /SCHEDULE=SIZE, is in effect, shorter jobs are printed before longer ones.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEARCH="search-string"

Resumes printing the current file of the current job on the first page containing the specified string. The string can be from 1 through 63 characters and must be enclosed in quotation marks. Use this qualifier in restarting an output queue from a paused state.

/SEPARATE=(option[,...])
/NOSEPARATE

Specifies the job separation defaults for a printer or terminal queue. The /SEPARATE qualifier cannot be used with the /GENERIC qualifier. The job separation options are as follows:

[NO]BURST	Specifies whether a burst page prints at the beginning of every job. Specifying BURST also results in a flag page being printed.
[NO]FLAG	Specifies whether a flag page prints at the beginning of every job.
[NO]TRAILER	Specifies whether a trailer page prints at the end of every job.
[NO]RESET=(module[,...])	Specifies a job reset sequence for the queue. The specified modules from the device control library are used to reset the device each time a job reset occurs.

/TERMINAL
/NOTERMINAL

Indicates that the output queue is a terminal queue. The /NOTERMINAL qualifier cancels the effect of a previous /TERMINAL qualifier on the same command. It is supported in this release for compatibility with VMS Version 4.n and may be retired in the future. The function of the /[NO]TERMINAL qualifier has been incorporated into the /[NO]DEVICE qualifier of the INITIALIZE/QUEUE command. DIGITAL recommends that you use this command to determine queue type and that existing command procedures using START/QUEUE/[NO]TERMINAL be updated.

/TOP_OF_FILE

Resumes printing at the beginning of the file that was current when the queue paused. Use this qualifier only when restarting an output queue from a paused state.

/WSDEFAULT=n

Defines a working set default for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values are a positive integer in the range 1 through 65,535, 0, or the word NONE can be specified for n. If you specify 0 or NONE, the working set default value becomes the value specified either in the UAF or by the SUBMIT command (if specified).

/WSEXTENT=n

Defines a working set extent for the batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values for n are a positive integer in the range 1 through 65,535, 0, or the word NONE. If you specify 0 or NONE, the working set value becomes the value specified either in the UAF or by the SUBMIT command (if specified).

/WSQUOTA=n

Defines the working set page size (working set quota) for a batch job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. Possible values are: a positive integer in the range 1 through 65,535, 0, or the word NONE as the value for n. If 0 or NONE is specified for n, the working set quota defaults to the value specified either in the UAF or by the SUBMIT command (if specified).

example

```
$ STOP/QUEUE  LPA0
$ START/QUEUE/TOP_OF_FILE  LPA0
```

The STOP/QUEUE command in this example suspends operation of the printer queue LPA0. Then the START/QUEUE/TOP_OF_FILE command resumes operation. The file that was being printed when the queue was stopped is started again from the beginning.

START/QUEUE/MANAGER

Starts the queue manager for the batch/print facility and opens the job queue manager file. After the system is bootstrapped, you must execute this command before you can execute any other queue management or job submission command. The /QUEUE qualifier is optional, but you must specify the /MANAGER qualifier.

Requires both OPER and SYSNAM privileges.

format

START/QUEUE/MANAGER *[file-spec]*

parameter

file-spec

Specifies the name of the file to contain information about batch and print jobs, queues, and form definitions. The file specification parameter is used in VAXcluster systems or for specifying an alternate system job queue file. The default file specification is SYS$SYSTEM:JBCSYSQUE.DAT. Any elements that you omit from the file specification default to those of SYS$SYSTEM:JBCSYSQUE.DAT. No wildcard characters are permitted in the file specification.

qualifiers

/BUFFER_COUNT=n

Specifies the number of buffers in a local buffer cache to allocate for performing I/O operations to the system job queue file. Specify a positive integer in the range of 1 through 127, or 0. If 0 is specified, the default value of 50 is used.

/EXTEND_QUANTITY=n

Specifies the number of blocks by which the system job queue file is extended, when necessary. This value is also used as the initial allocation size when the queue file is created. Specify a positive integer in the range of 10 through 65,535, or 0. If 0 is specified, the default value of 100 is used.

/NEW_VERSION
/NONEW_VERSION (default)

Specifies that a new version of the job queue manager file be created to supersede an existing version. All jobs in the previous version are lost if a new version is specified. The new file contains no information until you enter a subsequent INITIALIZE/QUEUE command.

/RESTART
/NORESTART (default)

The /RESTART qualifier specifies that the queue manager be restarted automatically on recovery from a job controller abort. In addition, batch and output queues are restored to the states that existed prior to the interruption of service. The job queue manager file that is opened is the same file that was open before the abort. Upon restarting, the job controller uses the default values for the /EXTEND_QUANTITY and /BUFFER_COUNT qualifiers. Previously set values are lost. When the job controller incurs an internal fatal error, the process aborts and restarts itself. By default, the queue manager is not restarted. Intervention by a user with OPERATOR privilege is necessary to restart the queue manager and to restore the queueing environment using START/QUEUE/MANAGER and appropriate START/QUEUE commands. Note that in order to prevent a looping condition, the job controller does not restart the queue manager if it detects an error within two minutes of starting the queue manager.

example

```
$ START/QUEUE/MANAGER DUA5:[SYSQUE]
```

The START/QUEUE/MANAGER command in this example opens the job queue manager file JBCSYSQUE.DAT on the cluster-accessible disk volume DUA5, in directory SYSQUE. You must mount the disk before you enter the START/QUEUE/MANAGER command.

STOP

Terminates execution of a command, an image, a command procedure, a command procedure that was interrupted by CTRL/Y, or a detached process or subprocess.

Requires GROUP privilege to stop other processes in the same group. Requires WORLD privilege to stop processes outside your group.

format

STOP *[process-name]*

parameter

process-name
Requires that the process be in your group.

Specifies the name of the process to be deleted. The process name can have from 1 to 15 alphanumeric characters. The specified process must have the same group number in its user identification code (UIC) as the current process; you cannot use the process-name parameter to stop a process outside of your group. To stop a process outside of your group, you must use the qualifier /IDENTIFICATION=pid.

qualifier

/IDENTIFICATION=pid
Specifies the system-assigned process identification code (PID). /IDENTIFICATION can be used in place of the process name parameter.

example

```
$ RUN/PROCESS_NAME=LIBRA  LIBRA
%RUN-S-PROC_ID, identification of created process is 0013340D
    .
    .
    .
$ STOP LIBRA
```

The RUN command in this example creates a subprocess named LIBRA to execute the image LIBRA.EXE. Subsequently, the STOP command causes the image to exit and deletes the process.

STOP/CPU

Stops the specified secondary processor or processors in a VMS multiprocessing system. The /CPU qualifier is required.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

STOP/CPU *[cpu-id,...]*

parameter

cpu-id

Decimal value representing the identity of a processor in a VMS multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0. If you do not specify a CPU ID, the STOP/CPU command selects a processor in the current active set to stop.

description

The STOP/CPU command removes a secondary processor from the active set in a VMS multiprocessing system. If the secondary processor is not executing a process when the STOP/CPU command is issued, it enters the STOPPED state. If the secondary is executing a process at the time, it continues to execute the current process until it becomes a candidate for rescheduling on another processor in the system. When this occurs, the secondary enters the STOPPED state.

The VMS operating system subjects a processor to a set of checks when it is the object of a STOP/CPU command. As a result, you may not be permitted to stop certain processors that are vital to the functioning of the system. In these cases, there is usually a process in the system that can execute only on the processor you intend to stop. You can determine this by issuing a SHOW CPU/FULL command. In unusual circumstances, you can bypass the checking mechanism by using the /OVERRIDE_CHECKS qualifier in the command.

The STOP/CPU command has no effect if its object processor is already in the STOPPED state when it is issued.

qualifiers

/ALL

Stops all eligible secondary processors in the system's active set.

/OVERRIDE_CHECKS
Directs the STOP/CPU command to bypass a series of checks that determine whether the specified processor is eligible for removal from the active set.

STOP/QUEUE

The STOP/QUEUE command causes the specified execution queue to pause. All jobs currently executing in the queue are suspended (until the queue is restarted with the START/QUEUE command), and no new jobs can be initiated. The /QUEUE qualifier is required.

Requires OPER privilege or EXECUTE (E) access to the queue.

format

STOP/QUEUE *queue-name[:]*

parameter

queue-name[:]
Specifies the name of the queue that you want to pause.

example

```
$ STOP/QUEUE TEXTBATCH
.
.
.
$ START/QUEUE/BLOCK_LIMIT=500 TEXTBATCH
```

The STOP/QUEUE command in this example halts all batch jobs that are currently executing on the queue TEXTBATCH and places that queue in the paused state. Later the START/QUEUE command releases the queue from the paused state. All the jobs that were halted resume processing, but the START/QUEUE command now limits any further jobs to 500 blocks or smaller.

STOP/QUEUE/ABORT

Aborts a job executing on a print or terminal queue, deletes it from the queue, and resumes execution of the other jobs in the queue. The /QUEUE qualifier is optional, but you must specify the /ABORT qualifier.

Requires OPER privilege, EXECUTE (E) access to the queue, or DELETE (D) access to the current job.

format

STOP/QUEUE/ABORT *queue-name[:]*

parameter

queue-name
Specifies the name of the queue containing the job you want to stop.

example

```
$ STOP/QUEUE/ABORT LPA0
```

This example aborts the current print job on the queue LPA0. The next pending job in the queue begins to execute. Assuming there is no problem with the printer, the current page of the file completes printing. If the printer queue has been set up to output trailer pages, a trailer page is printed before the job is suspended.

STOP/QUEUE/ENTRY

Stops the currently executing job on the specified batch queue and resumes execution of the next pending job in the queue. The entry number is the number assigned to the job when it is submitted to the queue. (Use the DELETE/ENTRY command to stop an entry that is queued and awaiting execution.) The /QUEUE qualifier is optional, but you must specify the /ENTRY qualifier.

Requires OPER privilege, EXECUTE (E) access to the queue, or DELETE (D) access to the current job.

format

STOP/QUEUE/ENTRY=*entry-number queue-name[:]*

parameter

queue-name
Specifies the name of the batch queue that contains the job you want to stop.

example

```
$ STOP/QUEUE/ENTRY=365 SYS$BATCH
```

The STOP/QUEUE/ENTRY command in this example stops batch job number 365 currently executing on the SYS$BATCH queue and begins the next pending job in the queue.

STOP/QUEUE/MANAGER

Performs an orderly shutdown of the system job queue manager on the node from which the command is entered. The /QUEUE qualifier is optional, but you must specify the /MANAGER qualifier.

Requires both OPER and SYSNAM privileges.

format

STOP/QUEUE/MANAGER

parameters

None.

example

```
$ STOP/QUEUE/MANAGER
```

The STOP/QUEUE/MANAGER command in this example performs a shutdown of all queues on the node from which the command is entered.

STOP/QUEUE/NEXT

Stops the specified queue after all executing jobs have completed processing. No new jobs can be initiated; the START/QUEUE command restarts the queue. The /QUEUE qualifier is optional, but you must specify the /NEXT qualifier.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

STOP/QUEUE/NEXT *queue-name[:]*

parameter

queue-name[:]
Specifies the name of the queue that you want to stop.

example

```
$ STOP/QUEUE/NEXT LPA0
$ SHOW QUEUE/ALL LPA0
Printer queue LPA0
$ DELETE/QUEUE LPA0
```

This example shows how to delete the printer queue LPA0. First, the STOP /QUEUE/NEXT command is entered, which stops the printer after the current job is printed. Then the SHOW QUEUE/ALL command is entered to ensure that no jobs are pending in the queue. The screen display shows that no jobs are pending. Finally, the DELETE/QUEUE command is entered to delete the printer queue LPA0.

STOP/QUEUE/REQUEUE

Stops the current job on the specified queue and requeues it for later processing. The queue does not stop; execution of the next pending job resumes. Print jobs that have been checkpointed resume printing at the checkpoint. Batch jobs containing SET RESTART_VALUE commands run those portions of the job that have not successfully completed. The /QUEUE qualifier is optional, but you must specify the /REQUEUE qualifier. If you are requeueing a job on a batch queue, you must specify the /ENTRY qualifier.

Requires OPER privilege, EXECUTE access to the queue or DELETE access to the current job.

format

STOP/QUEUE/REQUEUE*[=queue-name] queue-name[:]*
STOP/QUEUE/ENTRY=*entry-number/REQUEUE*
[=queue-name] queue-name[:]

parameter

queue-name
Specifies the name of the queue that contains the job you want to stop. When you also specify a queue name as a parameter for the /REQUEUE qualifier, the job is requeued to that queue.

qualifiers

/ENTRY=entry-number
Used with batch queues to stop a currently executing batch job. The entry-number is the job entry number that was assigned to the job when it was submitted to the queue. The job entry number that you specify must match the job entry number of an executing job in order for the STOP/QUEUE /REQUEUE/ENTRY command to take effect. You can only specify one entry number for each STOP/QUEUE/REQUEUE/ENTRY command.

/HOLD
Places the aborted job in a hold state for later release with the SET/QUEUE /ENTRY/RELEASE or SET QUEUE/ENTRY/NOHOLD command. (Use DELETE/ENTRY to delete a job in the hold state.)

/PRIORITY=n
Requires OPER or ALTPRI privileges to raise the priority value above the value of the SYSGEN parameter MAXQUEPRI. Changes the priority of the requeued job. The n parameter can be from 0 to 255; the default value of the n parameter is the same as the priority value that the job had when it was stopped.

example

```
$ STOP/QUEUE/REQUEUE=LPB0 LPA0
```

In this example, the current print job on queue LPA0 is stopped and requeued to queue LPB0. If the print job has been checkpointed, printing resumes on LPB0 where the job stopped on LPA0.

STOP/QUEUE/RESET

Abruptly stops the queue and returns control to the system. Any jobs currently executing are stopped immediately. The START/QUEUE command restarts the queue. Current jobs that can be restarted (all print jobs and any batch jobs submitted with the /RESTART qualifier) are requeued for processing. Current jobs that cannot be restarted are aborted and must be resubmitted for processing. The /QUEUE qualifier is optional, but you must specify the /RESET qualifier.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

STOP/QUEUE/RESET *queue-name[:]*

parameter

queue-name[:]
Specifies the name of the queue you want to reset.

example

```
$ STOP/QUEUE/RESET TEXBATCH
```

The STOP/QUEUE/RESET command in this example stops the TEXBATCH queue. Any current job that was submitted with the /RESTART qualifier is requeued for processing when the queue is restarted. Current jobs that did not specify /RESTART must be resubmitted to the queue.

SUBMIT

Enters one or more command procedures in a batch job queue.

Requires EXECUTE (E) access to the queue, WRITE (W) access to the queue, or OPER privilege. If you include a node name with the file specification parameter, you must use the /REMOTE qualifier.

format

SUBMIT *file-spec[,...]*

parameter

file-spec[,...]

Specifies the name of a file containing a command procedure. Wildcard characters are allowed. The default file type is COM. If a node name is specified, the /REMOTE qualifier must also be specified.

qualifiers

/AFTER=time
/NOAFTER

Requests that the job be held until after a specific time. If the specified time has already passed, the job is processed immediately. Time can be specified as either an absolute time or as a combination of absolute and delta times.

/BACKUP
/NOBACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]
/NOBEFORE

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]
/NOBY_OWNER

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CHARACTERISTICS=(characteristic[,...])

Specifies one or more characteristics that the execution queue must possess for the job to run. The characteristics you specify must be a subset of the characteristics associated with the queue that executes the job. Otherwise, the job remains pending until the queue characteristics are changed, or until you delete the entry with the DELETE/ENTRY command. By default, the job runs if no characteristics are specified.

/CLI=filename

Specifies the command language interpreter (CLI) to be used to process the job. The file specification assumes the device name SYS$SYSTEM: and the file type EXE (SYS$SYSTEM:filename.EXE).

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each SUBMIT operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET> . QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CPUTIME=keyword

Defines a CPU time limit for the batch job. Time can be specified as delta time, 0, NONE, or INFINITE. Both the value 0 and the keyword INFINITE allow unlimited CPU time; the keyword NONE defaults to your user authorization file (UAF) value or to the limit specified on the queue. Note that you cannot request more time than permitted by the base queue limits or by your own UAF.

/CREATED (default)
/NOCREATED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/DELETE
/NODELETE (default)

Positional qualifier. Controls whether files are deleted after processing. If you specify the /DELETE qualifier after the SUBMIT command name, all files in the job are deleted after processing. If you specify the /DELETE qualifier after a file specification, only that file is deleted after it is processed.

/EXCLUDE=(file-spec[,...])
/NOEXCLUDE

Excludes the specified files from the SUBMIT operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
/NOEXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/HOLD
/NOHOLD (default)

Controls whether or not the job is made available for immediate processing. The /HOLD qualifier holds the job until it is released by the /NOHOLD qualifier or by the /RELEASE qualifier of the SET QUEUE/ENTRY command.

/IDENTIFY (default)
/NOIDENTIFY

Displays the queue name and job-entry number of the job when it is queued.

/KEEP
/NOKEEP

Controls whether the log file is deleted after it is printed; /NOKEEP is the default unless /NOPRINTER is specified.

/LOG_FILE[=file-spec]
/NOLOG_FILE

Names the log file. The default log file name is job_name.LOG. No wildcards are allowed in the file specification. You can use the /LOG_FILE qualifier to write the log file to a different device. Logical names in the file specification are translated in the context of the process that submits the job.

/MODIFIED
/NOMODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED,

and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/NAME=job-name
Names the job (and possibly the batch job log file). The job name must be 1 to 39 alphanumeric characters. If characters other than alphanumerics, underscores, or dollar signs are used in the name, enclose the name in quotation marks. The default job name is the name of the first (or only) file in the job.

/NOTIFY
/NONOTIFY (default)
Controls whether a message is broadcast to your terminal when the job is completed or aborted.

/PARAMETERS=(parameter[,...])
Provides the values of up to 8 optional parameters (equated to the symbols P1 through P8, respectively, for each command procedure in the job). The symbols are local to the specified command procedure. If the parameter contains spaces, special characters, or lowercase characters, enclose it in quotation marks. The size of the parameter can be from 1 to 255 characters.

/PRINTER[=queue-name](default)
/NOPRINTER
Queues the job log file for printing when your job is completed. /PRINTER allows you to specify a particular print queue; the default print queue is SYS$PRINT. If you specify /NOPRINTER, /KEEP is assumed.

/PRIORITY=n
Requires OPER or ALTPRI to specify a priority greater than the value of the SYSGEN parameter MAXQUEPRI. Specifies the job-scheduling priority for the batch job with respect to other jobs in the same queue. The value of *n* is an integer in the range of 0 through 255. The default value is the value of the SYSGEN parameter DEFQUEPRI.

/QUEUE=queue-name[:]
Identifies the batch queue on which the job is entered. The default queue is SYS$BATCH.

/REMOTE
Indicates that the file resides on the remote node indicated and executes on the remote node. When using /REMOTE, the node name *must* be included in the file specification. Only the following qualifiers (which affect file selection) may be specified with /REMOTE: /BACKUP, /BEFORE, /BY_OWNER, /CONFIRM, /CREATED, /EXCLUDE, /EXPIRED, /MODIFIED, and /SINCE.

/RESTART
/NORESTART (default)

Indicates whether or not the job restarts after a system failure or after a STOP/QUEUE/REQUEUE command.

/SINCE[=time]
/NOSINCE

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/USER=username

Requires CMKRNL privilege and R (read) access to the user authorization file (UAF). Allows you to submit a job on behalf of another user. The job runs exactly as if that user had submitted it.

/WSDEFAULT=n

Defines a working set default for a batch job; the /WSDEFAULT qualifier overrides the working set size authorized by the system manager or specified in the user authorization file. Possible values of *n* are a positive integer in the range 1 through 65,535, 0, or the keyword NONE. Specify 0 or NONE if you want the working set value to default to either your UAF value or the working set default specified on the queue. You cannot request a higher value than your default.

/WSEXTENT=n

Defines a working set extent for the batch job; the /WSEXTENT qualifier overrides the working set extent authorized by the system manager or specified in the user authorization file. Possible values of *n* are a positive integer in the range 1 through 65,535, 0, or the keyword NONE. Specify 0 or NONE if you want the working set extent to default to either your UAF value or the working set extent specified on the queue. You cannot request a higher value than your default.

/WSQUOTA=n

Defines a working set page size (working set quota) for the batch job; the /WSQUOTA qualifier overrides the value established by the system manager or the value authorized in the user authorization file. Possible values of *n* are a positive integer in the range 1 through 65,535, 0, or the keyword NONE. Specify 0 or NONE if you want the working set quota to default to either your UAF value or the working set quota specified on the queue. You cannot request a higher value than your default.

example

```
$ DEFINE JUNE WORKZ:[JONES]ANNUAL_REPORT.COM
$ SUBMIT JUNE
Job JUNE (queue SYS$BATCH, entry 229) started on ZOO_BATCH
```

In this example, the logical name JUNE is created and equated to ANNUAL_REPORT.COM with the DEFINE command. Using the logical name JUNE, the user submits ANNUAL_REPORT.COM to the batch queue. Note that the system translates the logical name JUNE to ANNUAL_REPORT.COM before ANNUAL_REPORT.COM is submitted to the batch queue. Also, the log file produced is named ANNUAL_REPORT.COM rather than JUNE.COM.

Note also that the job is submitted to the generic queue SYS$BATCH, but runs on the execution queue ZOO_BATCH.

SYNCHRONIZE

Holds the process issuing the command until the specified batch job completes execution.

format

SYNCHRONIZE *[job-name]*

parameter

job-name
Specifies the name of the job as specified in the SUBMIT command. If you have more than one job with the same name, the process is synchronized with the last job submitted. To specify a job that does not have a unique name, use the /ENTRY qualifier to specify the job entry number. If you use the /ENTRY qualifier and if you also specify a job name, the job name is ignored.

qualifiers

/ENTRY=entry-number
Identifies the job by the system-assigned entry number. You must specify either the job name or the /ENTRY qualifier. If you specify both the job-name parameter and the /ENTRY qualifier, the job name is ignored.

/QUEUE=queue-name[:]
Names the queue containing the job. The default is SYS$BATCH.

example

```
$ SUBMIT/NAME=TIMER        COMP.COM
  Job TIMER (queue SYS$BATCH, entry 214) started on queue SYS$BATCH
$ SYNCHRONIZE /ENTRY=214
```

In this example, a batch job named TIMER is submitted. Then the SYNCHRONIZE command is entered interactively. This command places the interactive process in a wait state until entry number 214 (TIMER) completes. You cannot enter subsequent commands from your terminal until the SYNCHRONIZE command completes and your process is released from the wait state.

TYPE

Displays the contents of a file or group of files on the current output device.

format

TYPE *file-spec[,...]*

parameter

file-spec[,...]

Specifies one or more files to be displayed. If you specify a file name and not a file type, the file type defaults to LIS. The TYPE command displays all files that satisfy the file description.

Wildcard characters are allowed in place of the directory name, file name, file type, or file version number field. Either commas or plus signs can be used to separate two or more files. The files are displayed in the order listed.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each TYPE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET> . QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the TYPE operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you specify /OUTPUT=file-spec, the output is sent to the specified file, rather than to the current output device, SYS$OUTPUT.

/PAGE
/NOPAGE (default)

Controls whether output from the TYPE command is displayed one screen at a time.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

example

```
$ TYPE LETTER*.MEM
December 31, 1988
    .
    .
    .
CTRL/Y
Interrupt
$ SHOW TIME
 31-DEC-1988 15:48:07
$ CONTINUE
Sincerely yours,
    .
    .
    .
```

In this example, the TYPE command displays all files whose names begin with the word LETTER and have the file type MEM. While the files are being displayed, the user presses CTRL/Y to interrupt the TYPE operation and display the time. After entering the SHOW TIME command, the user enters the CONTINUE command to resume the TYPE operation.

UNLOCK

Makes an improperly closed file accessible.

format

UNLOCK *file-spec[,...]*

parameter

file-spec[,...]
Specifies the name of the file to be unlocked. Wildcard characters are allowed. If you include two or more file specifications, separate them with either commas or plus signs.

qualifiers

/CONFIRM
/NOCONFIRM (default)
For each file being unlocked, displays a query to which you must respond Y (YES) or T (TRUE) to unlock the file. Any other response aborts the unlock operation.

/LOG
/NOLOG (default)
Controls whether the UNLOCK command displays the file specification of each file being unlocked.

example

```
$ TYPE TST.OUT
%TYPE-E-OPENIN, error opening  DISK1:[STEVE]TST.OUT;3 as input
-SYSTEM-W-FILELOCKED, file is deaccess locked
$ UNLOCK TST.OUT
$ TYPE TST.OUT
```

In this example, the request to type the output file TST.OUT returns an error message indicating that the file is locked. The UNLOCK command unlocks it. Then the TYPE command is reentered to display the contents of the file.

WAIT

Puts your process into a wait state for the specified amount of time. The WAIT command is used in a command procedure to delay processing of either the procedure itself or a set of commands in the procedure.

format

WAIT *delta-time*

parameter

delta-time

Specifies a delta time interval in the following format. (A delta time is an offset from the current time to a time in the future.)

```
hour:minute:second.hundredth
```

The fields on the format line indicate the following:

hour	An integer in the range 0 through 59
minute	An integer in the range 0 through 59
second	An integer in the range 0 through 59
hundredth	An integer in the range 0 through 99.

The colons and period are required delimiters; also, the delta time must begin with the number of hours and not a colon. Note that the days field, usually included in the delta time format, must be omitted here.

example

```
$ LOOP:
$ RUN ALPHA
$ WAIT 00:10
$ GOTO LOOP
```

In this example, the command procedure executes the program image ALPHA. After the RUN command executes the program, the WAIT command delays execution of the GOTO command for 10 minutes. Note that 00 is specified for the number of hours, because the time specification cannot begin with a colon. After 10 minutes, the GOTO command executes, and the procedure transfers control to the label LOOP and executes the program ALPHA again. The procedure loops until it is interrupted or terminated.

If the procedure is executed interactively, terminate it by pressing CTRL/C or CTRL/Y and entering the STOP command or another DCL command that runs a new image in the process. If the procedure is executed in a batch job, enter the DELETE/ENTRY command to terminate it.

WRITE

Writes the specified data as one record to an open file specified by a logical name.

All qualifiers must precede all *data-item* expressions.

format

WRITE *logical-name expression[,...]*

parameters

logical-name
Specifies the logical name assigned to the output file. Use the logical name assigned by the OPEN command. In interactive mode, specify the process-permanent files identified by the logical names SYS$INPUT, SYS$OUTPUT, SYS$ERROR, and SYS$COMMAND.

expression[,...]
Specifies data to be written as a single record to the output file. You can specify data items using character string expressions, which may be symbol names, character strings in quotation marks, literal numeric values, or a lexical function. You can specify a list of expressions separated by commas; the command interpreter concatenates the items into one record and writes the record to the output file. The maximum size of any record that can be written is less than 1024 bytes. If, however, you specify the /SYMBOL qualifier, the maximum record size is 2048 bytes.

qualifiers

/ERROR=label
Transfers control on an I/O error to the location specified by *label* (in a command procedure). If no error routine is specified and an error occurs during the writing of the file, the current ON condition action is taken. /ERROR overrides any ON condition action specified. If an error occurs and control passes successfully to the target label, the reserved global symbol $STATUS retains the error code.

/SYMBOL
Causes the expression to be interpreted and its expanded value placed in a 2048-byte (instead of a 1024-byte) buffer before the write operation is performed. If you specify multiple expressions, their values are concatenated and placed in the 2048-byte buffer. Use the /SYMBOL qualifier to write a very large record. Each expression specified must be a symbol.

/UPDATE
Replaces the last record read with the record specified with the expression parameter. You must be able to read and write to a file to use the /UPDATE qualifier. Use the WRITE/UPDATE command only after a READ command. The WRITE/UPDATE command modifies the last record you have read.

example

```
$ OPEN/WRITE OUTPUT_FILE TESTFILE.DAT
$ INQUIRE ID "Assign Test-id Number"
$ WRITE/ERROR=WRITE_ERROR  OUTPUT_FILE  "Test-id is ",ID
$ WRITE/ERROR=WRITE_ERROR  OUTPUT_FILE  ""
$ !
$ WRITE_LOOP:
    .
    .
    .
$ GOTO WRITE_LOOP
$ END_LOOP:
$ !
$ CLOSE OUTPUT_FILE
$ PRINT TESTFILE.DAT
$ EXIT
$ !
$ WRITE_ERROR:
$ WRITE SYS$OUTPUT "There was a WRITE error."
$ CLOSE OUTPUT_FILE
$ EXIT
```

In this example, the open command opens the file TESTFILE.DAT; the INQUIRE command requests an identification number to be assigned to a particular run of the procedure. The number entered is equated to the symbol ID. The WRITE commands write a text line concatenated with the symbol name ID and a blank line.

The lines between the label WRITE_LOOP and END_LOOP process information and write additional data to the file. When the processing is finished, control is transferred to the label END_LOOP. The CLOSE and PRINT commands at this label close the output file and queue a copy of the file to the system printer.

The label WRITE_ERROR is used as the target of the /ERROR qualifier on the WRITE command; if an error occurs when a record is being written, control is transferred to the label WRITE_ERROR.

DIGITAL Standard Runoff (DSR) Commands

Digital Standard Runoff (DSR) creates formatted files from input files consisting of text, DSR commands, and DSR flags. The DCL commands RUNOFF, RUNOFF /CONTENTS, and RUNOFF/INDEX allow you to process DSR files to produce formatted output. See the description of the RUNOFF, RUNOFF/CONTENTS, and RUNOFF/INDEX in the Reference Section for more information on using these DCL commands.

The DSR commands in the DSR file determine how the text will be formatted in the output file. The format of a DSR command is as follows. The first character of a command must be the control flag, which is a period by default.

```
#.command-name [parameter,...][;]
```

You must observe the following rules:

- Period in column 1—The control flag (a period by default) of the first command on a line must be in column 1. No text can precede a command on a line.
- Multiple commands on one line—You can place as many DSR commands on one line as space permits except that commands taking text parameters may not be followed by another command.
- Text on a command line—You can follow a command with text if you terminate the command with a semicolon. The text must immediately follow the semicolon.
- Comments—You can put a comment in your DSR file wherever you can put a command, that is, as the first item on a line or one of a string of commands on a line. Format a comment as follows. The first character of a comment must be the comment flag, which is an exclamation point by default.

  ```
  !Any text except a semicolon[;]
  ```

- Separators—You must separate the command name from the first parameter and parameters from one another if two consecutive entries are both alphabetic or both numeric. Otherwise, you can choose to separate or not separate the entities. Valid separators between commands and parameters are spaces and tabs. Valid separators between parameters are spaces, tabs, and commas.
- Abbreviations—Abbreviations exist for the DSR command names. The abbreviations are listed in parentheses after the command names in the command descriptions that follow.
- Case—Command names can be in uppercase or lowercase.
- Null parameters—You can enter a null value for a parameter by typing just a comma.

.APPENDIX—(.AX) [title]

Starts an appendix by performing the following actions:

1. Issues the commands:

```
#.BREAK
#.LEFT MARGIN 0
#.SPACING 1
#.FILL (if .AUTOJUSTIFY is in effect)
#.JUSTIFY (if .AUTOJUSTIFY is in effect)
#.PAGING
#.PAGE
```

2. Inserts 12 blank lines.
3. Centers on the next line the word APPENDIX followed by a space and the letter or name identifying the appendix.
4. Inserts one blank line.
5. Centers on the next line the specified title. The title is written in uppercase unless case flags indicate otherwise.
6. Inserts three blank lines.

The appendix identifier is regulated by the .NUMBER APPENDIX and .DISPLAY APPENDIX commands.

The title becomes the running head. Any subtitle in effect is blanked.

.AUTOJUSTIFY—(.AJ)

Causes the following commands to issue .JUSTIFY and .FILL commands:

```
#.APPENDIX
#.CHAPTER
#.HEADER LEVEL
#.NOTE
```

Initial default.

.AUTOPARAGRAPH—(.AP)

Causes a .PARAGRAPH command to be issued whenever a line begins with a space or a tab. Cancels .AUTOTABLE if it is in effect. The .FILL command must be in effect.

.AUTOSUBTITLE—(.AST) [header-level]

Causes .HEADER LEVEL titles to be used for running head subtitles. *Header-level* specifies the highest header level for which subtitles will be written and takes one of the following forms:

- Integer in the range 1 through 6—The exact header level.

- Integer preceded by a plus sign—A value to be added to the last header level specified in an .AUTOSUBTITLE command.
- Integer preceded by a minus sign—A value to be subtracted from the last header level specified in an .AUTOSUBTITLE command.

Header-level defaults to 1.

You must issue a .SUBTITLE command for .AUTOSUBTITLE to work. See .SUBTITLE for other effects connected with subtitles.

Initial default: .AUTOSUBTITLE 1

.AUTOTABLE—(.AT)

Causes a .PARAGRAPH command to be issued whenever a line does not begin with a space or a tab. Cancels .AUTOPARAGRAPH if it is in effect. The .FILL command must be in effect.

.BEGIN BAR—(.BB)

Begins the insertion of change bars at the beginning of lines. The .ENABLE BAR command must be in effect.

.BLANK—(.B) [lines]

Issues a .BREAK command and inserts blank lines. Specify *lines* as either:

- Zero or unsigned integer—The number of blank lines to be inserted.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Lines defaults to 1.

The .BLANK command does not work at the top of a page. (Use the .FIGURE command.) The .BLANK command does not continue to the next page if *lines* is greater than the number of lines left on the page. If the page contains a footnote, the line directly above a footnote is considered the bottom of the page.

.BREAK—(.BR)

Ends the current line without filling or justification. A .BREAK command immediately following a .PARAGRAPH, .INDENT, .LEFT MARGIN, .AUTOPARAGRAPH, or .AUTOTABLE command cancels any specified indentation.

.CENTER—(.C) [line-size]

Issues a .BREAK command and centers the text that follows. Specify *line-size* as follows:

- Unsigned integer—Twice the value of the position that you want to center the text around.
- Integer preceded by a plus sign—A value to be added to the last centering position used.
- Integer preceded by a minus sign—A value to be subtracted from the last centering position used.

Line-size defaults to the value of the left margin plus the value of the right margin, which causes the text to be centered between the two margins.

The .CENTER command must be the last command on a line. Enter the text to be centered on the next line or terminate the command with a semicolon and enter the text to be centered immediately after the semicolon. The text must all be on one line. The line can contain flags but not commands. The text can extend beyond margin and page size settings but cannot extend to the left of position 0.

.CHAPTER—(.CH) [title]

Starts a chapter by performing the following actions:

1. Issues the commands:

```
#.BREAK
#.LEFT MARGIN 0
#.SPACING 1
#.FILL (if .AUTOJUSTIFY is in effect)
#.JUSTIFY (if .AUTOJUSTIFY is in effect)
#.PAGING
#.PAGE
```

2. Inserts 12 blank lines.
3. Centers on the next line the word CHAPTER followed by a space and the number of the chapter.
4. Inserts one blank line.
5. Centers on the next line the specified title. The title is written in uppercase unless case flags indicate otherwise.
6. Inserts three blank lines.

The chapter number is regulated by the .NUMBER CHAPTER and .DISPLAY CHAPTER commands.

The title becomes the running head. Any subtitle in effect is blanked.

.COMMENT—(.!) [text]

Ignores the text for output processing.

.CONTROL CHARACTERS—(.CC)

Accepts and places in the output file nonprinting characters (characters with ASCII values in the ranges 0 through 31 and 127 through 160).

.DATE—(.D)

Adds the date to running heads. You must issue a .SUBTITLE command for .DATE to work. Either .LAYOUT 1 or .LAYOUT 2 overrides the .DATE command.

.DISABLE BAR—(.DBB)

Disables the .BEGIN BAR and .END BAR commands. If .ENABLE BAR has been in effect, .DISABLE BAR does not shift the lines of text back to their original positions.

Initial default.

.DISABLE BOLDING—(.DBO)

Disables use of the bold flag.

.DISABLE HYPHENATION—(.DHY)

Disables use of the hyphenation flag.

.DISABLE INDEXING—(.DIX)

Disables use of the index flag and the commands .INDEX and .ENTRY.

.DISABLE OVERSTRIKING—(.DOV)

Disables use of the overstrike flag.

.DISABLE TOC—(.DTC)

Disables the collection of information for a table of contents.

.DISABLE UNDERLINING—(.DUL)

Disables use of the underline flag.

.DISPLAY APPENDIX—(.DAX) format-code

Issues a .BREAK command and defines the appearance of appendix identifiers. Specify *format-code* as one of the following:

D	Decimal numbers
O	Octal numbers
H	Hexadecimal numbers
RU	Uppercase roman numerals
RL	Lowercase roman numerals
RM	Mixed case roman numerals (initial capitals)
LU	Uppercase letters
LL	Lowercase letters
LM	Mixed case letters (initial capitals)

Initial default: .DISPLAY APPENDIX LU

.DISPLAY CHAPTER—(.DCH) format-code

Issues a .BREAK command and defines the appearance of chapter identifiers. Specify *format-code* as described in the .DISPLAY APPENDIX command.

Initial default: .DISPLAY CHAPTER D

.DISPLAY ELEMENTS—(.DLE) ["left",]format-code[,"right"]

Issues a .BREAK command and defines the appearance of list-element identifiers. The command must be specified after the .LIST command and before the first .LIST ELEMENT command.

List-element identifiers consist of sequential numbers in the format specified by *format-code*, preceded by the character specified as *left* and followed by the character specified as *right*. *Left*, which must be a single character enclosed in quotation marks or apostrophes, defaults to a space. *Right*, which must be a single character enclosed in quotation marks or apostrophes, defaults to a period. Specify *format-code* as described in the .DISPLAY APPENDIX command.

A .DISPLAY ELEMENTS command remains in effect only until the .END LIST command occurs. The next list uses the default appearance unless another .DISPLAY ELEMENTS command is specified.

Default appearance of lists: .DISPLAY ELEMENTS "",D,"."

.DISPLAY LEVELS—(.DHL) [format-code],...

Issues a .BREAK command and defines the appearance of section identifiers in section headers. You can specify or omit up to six *format-code* values, one for each section level. The first *format-code* value corresponds to section level 1, the second *format-code* value to section level 2, and so on. If you omit a *format-code* value, specify the comma unless no more values follow. Specify each *format-code* value as described in the .DISPLAY APPENDIX command.

Initial default: .DISPLAY LEVELS D,D,D,D,D,D

.DISPLAY NUMBER—(.DNM) format-code

Issues a .BREAK command and defines the appearance of page numbers starting with the next page number written. Specify *format-code* as described in the .DISPLAY APPENDIX command.

Initial default: .DISPLAY NUMBER D

.DISPLAY SUBPAGE—(.DSP) format-code

Issues a .BREAK command and defines the appearance of subpage numbers starting with the next page number written. (Subpage numbers are the numbers appended to the page numbers on subpages, for example, page 1-12A.) Specify *format-code* as described in the .DISPLAY APPENDIX command.

Initial default: .DISPLAY SUBPAGE LU

.ELSE—variant

Starts the *else* portion of a conditional block. *Variant* must be the same value as in the .IF command or .IFNOT command. The *else* portion is processed if the *if* or *ifnot* portion of the conditional block is not processed; otherwise, the *else* portion is not processed. Must be paired with an .IF or .IFNOT command.

.ENABLE BAR—(.EBB)

Enables use of the .BEGIN BAR and .END BAR commands. Causes all further text to be shifted three characters to the right to make room for the bars.

.ENABLE BOLDING—(.EBO)

Enables use of the bold flag.

Initial default.

.ENABLE HYPHENATION—(.EHY)

Enables use of the hyphenation flag.

Initial default.

.ENABLE INDEXING—(.EIX)

Enables use of the index flag and the commands .INDEX and .ENTRY.

Initial default.

.ENABLE OVERSTRIKING—(.EOV)

Enables use of the overstrike flag.

Initial default.

.ENABLE TOC—(.ETC)

Enables the collection of information for a table of contents.

Initial default.

.ENABLE UNDERLINING—(.EUN)

Enables use of the underline flag.

Initial default.

.END BAR—(.EB)

Ends the insertion of change bars at the beginning of lines. Must be paired with a .BEGIN BAR command.

.END FOOTNOTE—(.EFN)

Ends a footnote and restores case, fill, justify, spacing, and margin settings to what they were before the footnote began. Must be paired with a .FOOTNOTE command.

.END LIST—(.ELS) [lines]

Ends a list and restores case, fill, justify, spacing, and margin settings to what they were before the list began. Must be paired with a .LIST command. Specify *lines* as one of the following:

- Zero or unsigned integer—The number of blank lines to follow the list.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Lines defaults to the most recent skip lines setting in a .PARAGRAPH or .SET PARAGRAPH command (parameter 2).

.END LITERAL—(.EL)

Ends literal text. Must be paired with a .LITERAL command.

.END NOTE—(.EN) [lines]

Ends a note and restores case, fill, justify, spacing, and margin settings to what they were before the note began. Must be paired with a .NOTE command. Specify *lines* as one of the following:

- Zero or unsigned integer—The number of blank lines to follow the note.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Lines defaults to a value of 1.

.END SUBPAGE—(.ES)

Issues the .BREAK command and begins a new page with normal page numbering.

.ENDIF—variant

Ends a conditional block. *Variant* must be the same value as in the .IF or .IFNOT command. Must be paired with an .IF or .IFNOT command.

.ENTRY—(.Y) topic[> subtopic]...

Creates an index entry without a page reference. The parameters specify the text of the entries. Subtopics are arranged alphabetically under topics.

.FIGURE—(.FG) [lines]

Issues the .BREAK command and inserts the number of blank lines specified by lines. If the current page does not have sufficient room for all the blank lines, the page is ended and the blank lines are placed on the next page.

Lines must be specified as an integer in the range 1 through the maximum number of lines permitted on the page (after header, footer, and forced blank lines are taken into account). *Lines* defaults to a value of 1.

.FIGURE DEFERRED—(.FGD)[lines]

Issues the .BREAK command and inserts the number of blank lines specified by lines. If the current page does not have sufficient room for all the blank lines, text is added until the page is complete and the blank lines are placed on the next page.

Lines must be specified as an integer in the range 1 through the maximum number of lines permitted on the page (after header, footer, and forced blank lines are taken into account). *Lines* defaults to a value of 1.

.FILL—(.F)

Causes line endings in the DSR file to be treated as spaces. Lines are created in the output file by accumulating words until the next word would exceed the right margin. The .FILL command also restores: the most recent justification setting set by a .JUSTIFY or .NO JUSTIFY command; any .AUTOPARAGRAPH or .AUTOTABLE setting that was in effect.

Initial default.

.FIRST TITLE—(.FT)

Permits a running head to be written on the first page unless a .CHAPTER or .APPENDIX command is issued.

.FLAGS ACCEPT—(.FL ACCEPT) [character]

Recognizes the accept flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the accept flag is represented by an underscore (_).

Initial default.

.FLAGS ALL—(.FL)

Permits recognition of all enabled flags.

Initial default.

.FLAGS BOLD—(.FL BOLD) [character]

Recognizes the bold flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the bold flag is represented by an asterisk (*).

.FLAGS BREAK—(.FL BREAK) [character]

Recognizes the break flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the break flag is represented by a vertical bar (|).

.FLAGS CAPITALIZE—(.FL CAPITALIZE) [character]

Recognizes the capitalize flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the capitalize flag is represented by a left angle bracket (<).

.FLAGS COMMENT—(.FL COMMENT) [character]

Recognizes the comment flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the comment flag is represented by an exclamation point (!).

Initial default.

.FLAGS CONTROL—(.FL CONTROL) [character]

Recognizes the control flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the control flag is represented by a period (.).

Initial default.

.FLAGS HYPHENATE—(.FL HYPHENATE) [character]

Recognizes the hyphenate flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the hyphenate flag is represented by an equal sign (=).

.FLAGS INDEX—(.FL INDEX) [character]

Recognizes the index flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the index flag is represented by a right angle bracket (>).

.FLAGS LOWERCASE—(.FL LOWERCASE) [character]

Recognizes the lowercase flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the lowercase flag is represented by a backslash (\).

Initial default.

.FLAGS OVERSTRIKE—(.FL OVERSTRIKE) [character]

Recognizes the overstrike flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the overstrike flag is represented by a percent sign (%).

.FLAGS PERIOD—(.FL PERIOD) [character]

Recognizes the period flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the period flag is represented by a plus sign (+).

.FLAGS SPACE—(.FL SPACE) [character]

Recognizes the space flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the space flag is represented by a number sign (#).

Initial default.

.FLAGS SUBINDEX—(.FL SUBINDEX) [character]

Recognizes the subindex flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the subindex flag is represented by a right angle bracket (>).

Initial default.

.FLAGS SUBSTITUTE—(.FL SUBSTITUTE) [character]

Recognizes the substitute flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the substitute flag is represented by a dollar sign ($).

.FLAGS UNDERLINE—(.FL UNDERLINE) [character]

Recognizes the underline flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the underline flag is represented by an ampersand (&).

Initial default.

.FLAGS UPPERCASE—(.FL UPPERCASE) [character]

Recognizes the uppercase flag and optionally specifies a character to represent it. *Character* must be specified as a single character. By default, the uppercase flag is represented by a circumflex (^).

Initial default.

.FOOTNOTE—(.FN)

Places the text that follows up to the .END FOOTNOTE command at the bottom of the page if it fits, or at the bottom of the next page if it does not fit. Must be paired with an .END FOOTNOTE command.

The .FOOTNOTE command does not provide any special formatting. Format the footnote by including DSR commands within the footnote.

If the .NO PAGING command is in effect, all footnotes appear at the end of the document.

.HEADER LEVEL—(.HL) [level] [title]

Starts a section by performing the following actions:

1. Issues the commands:

```
#.BREAK
#.TEST PAGE lines (as specified in .STYLE HEADER)
#.SPACING 1
#.FILL (if .AUTOJUSTIFY is in effect)
#.JUSTIFY (if .AUTOJUSTIFY is in effect)
```

2. Writes the section number and header, formatted according to the .STYLE HEADER values in effect.

3. Inserts punctuation and/or spacing according to the .STYLE HEADER values in effect.

Level must be specified as one of the following:

- Unsigned integer in the range 1 to 6—The exact level of the header.
- Integer preceded by a plus sign—A value to be added to the last level specified in a .HEADER LEVEL or .SET LEVEL command (or 1 if a level has not yet been set).
- Integer preceded by a minus sign—A value to be subtracted from the last level specified in a .HEADER LEVEL or .SET LEVEL command (or 1 if a level has not yet been set).

Level defaults to the value of the last level specified in a .HEADER LEVEL or .SET LEVEL command (or 1 if the level has not yet been set).

The title must follow on the same line as the command with no intervening semicolon. If the title exceeds the size of the output line, the first line is filled (and justified if justification is in effect) and the title continues on the next line.

The title becomes the running head. Any subtitle in effect is blanked.

.HEADERS LOWER—(.HD LOWER)

Writes in lowercase the word "page" that precedes the page number in some layouts. Writes in lowercase the word "index" that is part of the page number in indexes.

.HEADERS MIXED—(.HD MIXED)

Writes in lowercase with an initial capital the word "Page" that precedes the page number in some layouts. Writes in lowercase with an initial capital the word "Index" that is part of the page number in indexes.

Initial default.

.HEADERS [ON]—(.HD)

Writes running heads on each page except the first (unless you also specify .FIRST TITLE) according to the current .LAYOUT values.

Initial default.

.HEADERS UPPER—(.HD UPPER)

Writes in uppercase the word "PAGE" that precedes the page number in some layouts. Writes in uppercase the word "INDEX" that is part of the page number in indexes.

.IF—variant

Starts a conditional block and introduces the *if* portion of the block. The *if* portion of the block is processed if *variant* is specified as a value of the /VARIANT qualifier in the invoking DSR command. The .IF command must be paired with an .ENDIF command. The conditional block can contain an .ELSE command.

You can nest one conditional block within another. The nested conditional block must be entirely contained within the *if* or the *else* portion of the nesting block.

.IFNOT—variant

Starts a conditional block and introduces the *ifnot* portion of the block. The *ifnot* portion of the block is processed if *variant* is not specified as a value to the /VARIANT qualifier in the invoking DSR command. The .IFNOT command must be paired with an .ENDIF command and can be paired with an .ELSE command. The .ELSE command must precede the .ENDIF command.

You can nest one conditional block within another. The nested conditional block must be entirely contained within the *if* or the *else* portion of the nesting block.

.INDENT—(.I) spaces

Issues a .BREAK command and indents a line of text. If you specify a .BREAK command after .INDENT, the indent operation is canceled. Specify *spaces* as one of the following:

- Zero or unsigned integer—The number of spaces to indent to the right of the left margin.
- Integer preceded by a minus sign—The number of spaces to indent to the left of the left margin. You cannot indent past position 0.

The parameter defaults to the value of the *spaces* parameter of the most recent .PARAGRAPH or .SET PARAGRAPH command.

You can enter the line of text after the command (with an intervening semicolon) or on the next line. The line cannot contain commands.

.INDEX—(.X) topic[> subtopic...]

Creates an index entry with a page reference. The parameters specify the text of the entries. Subtopics are arranged alphabetically under topics with their own page references.

.JUSTIFY—(.J)

Issues a .BREAK command and causes the text that follows to be justified—extra spaces are inserted between words so that the last character on each line reaches the right margin.

Initial default.

.KEEP—(.K)

Causes blank lines in the input file to be inserted in the output file. The .NO FILL command must be in effect.

.LAYOUT—(.LO) code [,lines]

Issues a .BREAK command and specifies the format for page header and footer information. Specify *code* as one of the following:

Code	Description
0	Running head appears in the upper left of the page. Page number and date appear in the upper right.
1	Running head appears centered at the top of the page. Page number appears centered at the bottom.
2	Running head appears at the top right of an odd-numbered page and the top left of an even-numbered page. Page number appears centered at the bottom.
3	Running head appears in the upper left of the page. Date appears in the upper right. Page number appears centered between hyphens at the bottom of the page. Page numbers are consecutive through the entire document.

If code equals 1, 2, or 3, specify *lines* as an unsigned integer equal to the number of lines below the last line of text that the page number will appear. If code equals 0, do not specify *lines*.

Initial default: .LAYOUT 0

.LEFT MARGIN—(.LM) position

Issues a .BREAK command and sets the position of the left margin. Specify *position* as one of the following:

- Unsigned integer—The exact position of the left margin.
- Integer preceded by a plus sign—A value to be added to the current position of the left margin.

- Integer preceded by a minus sign—A value to be subtracted from the current position of the left margin.

Position defaults to a value of 0.

The left margin must be greater than 0 and less than the right margin.

Initial default: .LEFT MARGIN 0

.LIST—(.LS) [lines] [,"character"]

Starts a list by performing the following actions:

1. Issues a .BREAK command.
2. Issues the command .LEFT MARGIN 9 if the left margin is currently 0; otherwise, issues the command .LEFT MARGIN +4.
3. Issues a .TEST PAGE command specifying for the *lines* parameter a value of 2 plus the value of *test-lines* in the most recent .PARAGRAPH or .SET PARAGRAPH command.

Specify *lines* as one of the following:

- Zero or unsigned integer—The number of blank lines to be inserted before each element in the list.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Specify *character* as a character enclosed in quotation marks or apostrophes. Each list element is preceded by this character and two spaces. If you omit *character*, the list elements are preceded by a sequence of numbers starting with number 1.

The .LIST command must be paired with an .END LIST command. A list can contain .LIST ELEMENT commands. You can nest one list within another to a maximum level of 14; a nested list must be entirely contained within one element of the nesting list.

.LIST ELEMENT—(.LE)

Issues a .BREAK command and writes the text that follows as a list element. The text is terminated by either another .LIST ELEMENT command or an .END LIST command.

You can specify .LIST ELEMENT without having issued a .LIST command. List elements not enclosed in lists begin at the left margin and are numbered sequentially throughout the entire document. Do not issue an .END LIST command.

.LITERAL—(.LT)

Issues a .BREAK command, issues the command .RIGHT MARGIN 150, and writes text as it appears in the input file until an .END LITERAL command occurs. Must be paired with an .END LITERAL command. When .LITERAL is in effect, flags are treated as text even if they are recognized and enabled. Commands and flags in effect before the .LITERAL command are disabled (until the .END LITERAL command occurs) except for .LEFT MARGIN, .TAB STOPS, the underline flag, and the bold flag.

.NO AUTOJUSTIFY—(.NAJ)

Disables the effects of the .AUTOJUSTIFY command.

.NO AUTOPARAGRAPH—(.NAP)

Disables the effects of the .AUTOPARAGRAPH command.

Initial default.

.NO AUTOSUBTITLE—(.NAST)

Disables the effects of the .AUTOSUBTITLE command.

.NO AUTOTABLE—(.NAT)

Disables the effects of the .AUTOTABLE command.

Initial default.

.NO CONTROL CHARACTERS—(.NCC)

Disables the effects of the .CONTROL CHARACTERS command.

Initial default.

.NO DATE—(.ND)

Disables the effects of the .DATE command.

Initial default.

.NO FILL—(.NF)

Issues a .BREAK command and disables the effects of the .AUTOPARAGRAPH, .AUTOTABLE, .FILL, and .JUSTIFY commands.

.NO FLAGS ACCEPT—(.NFL ACCEPT)

Disables recognition of the accept flag.

.NO FLAGS [ALL]—(.NFL [ALL])

Does not permit recognition of flags except the comment and control flags.

Initial default.

.NO FLAGS BOLD—(.NFL BOLD)

Disables recognition of the bold flag.

Initial default.

.NO FLAGS BREAK—(.NFL BREAK)

Disables recognition of the break flag. **Initial default.**

.NO FLAGS CAPITALIZE—(.NFL CAPITALIZE)

Disables recognition of the capitalize flag.

Initial default.

.NO FLAGS COMMENT—(.NFL COMMENT)

Disables recognition of the comment flag.

.NO FLAGS CONTROL—(.NFL CONTROL)

Disables recognition of the control flag.

.NO FLAGS HYPHENATE—(.NFL HYPHENATE)

Disables recognition of the hyphenate flag.

Initial default.

.NO FLAGS INDEX—(.NFL INDEX)

Disables recognition of the index flag.

Initial default.

.NO FLAGS LOWERCASE—(.NFL LOWERCASE)

Disables recognition of the lowercase flag.

.NO FLAGS OVERSTRIKE—(.NFL OVERSTRIKE)

Disables recognition of the overstrike flag.

Initial default.

.NO FLAGS PERIOD—(.NFL PERIOD)

Disables recognition of the period flag.

Initial default.

.NO FLAGS SPACE—(.NFL SPACE)

Disables recognition of the space flag.

.NO FLAGS SUBINDEX—(.NFL SUBINDEX)

Disables recognition of the subindex flag.

.NO FLAGS SUBSTITUTE—(.NFL SUBSTITUTE)

Disables recognition of the substitute flag.

Initial default.

.NO FLAGS UNDERLINE—(.NFL UNDERLINE)

Disables recognition of the underline flag.

.NO FLAGS UPPERCASE—(.NFL UPPERCASE)

Disables recognition of the uppercase flag.

.NO JUSTIFY—(.NJ)

Issues a .BREAK command and disables the effects of the .JUSTIFY command.

.NO KEEP—(.NK)

Disables the effects of the .KEEP command.

Initial default.

.NO NUMBER—(.NNM)

Suspends the writing of page numbers, unless .LAYOUT 3 is in effect.

.NO PAGING—(.NPA)

Causes the document to be written without page breaks and without reserving room for headers and footers.

.NO PERIOD—(.NPR)

Disables the effects of the .PERIOD command.

.NO SPACE—(.NSP)

Causes a space not to be inserted in place of a carriage return between two lines of text. The .NO SPACE command must be placed between two lines of text and affects only those lines. The .FILL command must be in effect.

.NO SUBTITLE—(.NST)

Causes subtitles not to be written.

Initial default.

.NOTE—(.NT) [title]

Performs the following actions:

1. Issues a .BREAK command.
2. Issues a .TEST PAGE command specifying for the *lines* parameter the value of *test-lines* in the most recent .PARAGRAPH or .SET PARAGRAPH command.
3. Issues the command .SKIP 1, writes the specified title, and issues the command .SKIP 1. The title defaults to the word NOTE in uppercase.
4. If the left margin is set to position 0, issues the commands .LEFT MARGIN +8 and .RIGHT MARGIN –8. Otherwise, issues the commands LEFT MARGIN +4 and .RIGHT MARGIN –4.
5. Issues the command .FILL. Issues the command .JUSTIFY if autojustification is in effect. Writes the text that follows the .NOTE command until an .END NOTE command occurs.

A .NOTE command must be paired with an .END NOTE command.

.NUMBER APPENDIX—(.NMAX) [identifier]

Specifies the identifier of the next appendix as follows:

- Character—A single character.
- Character string—Up to five characters.
- Unsigned integer—The sequence number of the letter in the alphabet: 1 means A, 2 means B, 26 means Z, 27 means AA, and so on.
- Integer preceded by a plus sign—A value to be added to the sequence number of the current identifier.
- Integer preceded by a minus sign—A value to be subtracted from the sequence number of the current identifier.

Identifier defaults to a value of A.

If you do not explicitly provide an identification for the next appendix with the .NUMBER APPENDIX command, the identifier assumes the value of the identifier of the current appendix incremented by one.

Initial default: .NUMBER APPENDIX A

.NUMBER CHAPTER—(.NMCH) number

Specifies the identifier of the next chapter as follows:

- Unsigned integer—The number of the chapter.
- Integer preceded by a plus sign—A value to be added to the number of the current chapter.
- Integer preceded by a minus sign—A value to be subtracted from the number of the current chapter.

Number defaults to a value of 1.

If you do not explicitly provide an identification for the next chapter with the .NUMBER CHAPTER command, the chapter assumes the value of the current chapter number incremented by one.

Initial default: .NUMBER CHAPTER 1

.NUMBER LEVEL—(.NMLV) [number],...

Specifies a base section number. You can specify up to six numbers: each number corresponds to a level. Omit a level by including just the comma, except that trailing commas can be omitted. A number defaults to the current number in effect for the level. Specify each number as follows:

- Unsigned integer—The number of the section for the indicated level.
- Integer preceded by a plus sign—A value to be added to the current number for the indicated level.
- Integer preceded by a minus sign–A value to be subtracted from the current number for the indicated level.

If you do not explicitly provide a number for the next section header with the .NUMBER LEVEL command, the section header number assumes the value of the current section header number incremented by one.

.NUMBER LIST—(.NMLS) number

Specifies the number of the next element in a list.

If you do not explicitly provide a number for the next list element with the .NUMBER LIST command, the list element number assumes the value of the current list element number incremented by one.

.NUMBER [PAGE]—(.NMPG) [number]

Resumes page numbering if .NO NUMBER is in effect. Sets the number of the next page to the value of *number*, which must be one of the following:

- Unsigned integer—The number of the next page.
- Integer preceded by a plus sign—A value to be added to the current page number.
- Integer preceded by a minus sign—A value to be subtracted from the current page number.

Number defaults to the current page number.

Do not use the .NUMBER PAGE command if the .LAYOUT 3 command is in effect.

.NUMBER RUNNING—(.NMR) number

Sets the number of the next page to the value of *number*, which must be one of the following:

- Unsigned integer—The number of the next page.
- Integer preceded by a plus sign—A value to be added to the current page number.
- Integer preceded by a minus sign—A value to be subtracted from the current page number.

Number defaults to the current page number.

Use only if the .LAYOUT 3 command is in effect.

.NUMBER SUBPAGE—(.NMSPG) [identifier]

Specifies the identifier of the next subpage as follows:

- One or more characters—A letter such as A, B, Z, or AA.
- Unsigned integer—The sequence number of the letter in the alphabet: 1 means A, 2 means B, 26 means Z, 27 means AA, and so on.
- Integer preceded by a plus sign—A value to be added to the sequence number of the current identifier.
- Integer preceded by a minus sign—A value to be subtracted from the sequence number of the current identifier.

Identifier defaults to a value of A.

The .NUMBER SUBPAGE command issues a .SUBPAGE command if .SUBPAGE is not already in effect.

.PAGE—(.PG)

Issues a .BREAK command and starts a new page. The current page must contain at least one line of text. The .PAGE command works even if .NO PAGING is in effect.

.PAGE SIZE—(.PS) [max-lines], [running-width]

Issues a .BREAK command, sets a maximum for the number of lines of text on a page, and sets the page width for writing running heads (does not affect normal text). Specify *max-lines* as follows:

- Unsigned integer—The maximum number of lines allowed.
- Integer preceded by a plus sign—A value to be added to the current maximum.
- Integer preceded by a minus sign—A value to be subtracted from the current maximum.

Max-lines defaults to the current maximum. The maximum cannot be less than 13.

Specify *running-width* as follows:

- Unsigned integer—The width of the page for writing running heads.
- Integer preceded by a plus sign—A value to be added to the current width.
- Integer preceded by a minus sign—A value to be subtracted from the current width.

Running-width defaults to the current width. The width cannot exceed 150.

If the .NO PAGING command is in effect, the .PAGE SIZE command issues a .PAGING command.

Initial default: .PAGE SIZE 58,70

.PAGING—(.PA)

Issues a .BREAK command and causes the document to be split into pages.

Initial default.

.PARAGRAPH—(.P) [spaces],[skip-lines],[test-lines]

Issues a .BREAK command followed by .TEST PAGE, .SKIP, and .INDENT commands. Specify *spaces* as one of the following:

- Zero or unsigned integer—The number of spaces to indent the first line of the paragraph to the right of the left margin.
- Integer preceded by a minus sign—The number of spaces to indent to the left of the left margin. You cannot indent past position 0.

Specify *skip-lines* as one of the following:

- Zero or unsigned integer—The number of blank lines to be inserted before the paragraph.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Specify *test-lines* as a factor in determining the number of lines that must be left on the page for the paragraph to start on the page. Otherwise, the paragraph starts on the next page. The precise algorithm is as follows, where *spacing* is the current value of the .SPACING command (initial default is 1):

```
(skip-lines + test-lines + 1 ) * spacing
```

All three parameters default to the values specified in the last .PARAGRAPH or .SET PARAGRAPH command.

Initial default: .PARAGRAPH 5,1,2

.PERIOD—(.PR)

Adds an extra space after a period (.), colon (:), question mark (?), or exclamation point (!). The .FILL command must be in effect, the character must be followed by a space or carriage return, and the character must not be preceded by the accept flag.

Initial default.

.REPEAT—(.RPT) times, "characters"

Repeats the specified characters the specified number of times. The characters must be enclosed in quotation marks or apostrophes. You cannot specify more than 150 characters.

If the .FILL command is in effect, the characters are repeated horizontally. If the .NOFILL command is in effect, the characters are repeated vertically starting at the left margin.

.REQUIRE—(.REQ) "file-spec"

Processes the contents of another file as if those contents existed in place of the .REQUIRE command. The file specification must be enclosed in quotation marks or apostrophes. The file type defaults to RNO. A .REQUIRE command must be the last command on a line.

.RESTORE—(.RE)

Restores the formatting context saved by the last .SAVE command. Must be paired with a .SAVE command.

.RIGHT—(.R) [spaces]

Issues a .BREAK command and positions a line of text relative to the right margin. If you specify a .BREAK command after the .RIGHT, the operation is canceled. Specify *spaces* as one of the following:

- Zero or an unsigned integer—The number of spaces to indent to the left of the right margin. You cannot indent past position 0.
- Integer preceded by a minus sign—The number of spaces to extend to the right of the right margin.

Spaces defaults to 0.

You can enter the line of text after the command (with an intervening semicolon) or on the next line. The line cannot contain commands.

.RIGHT MARGIN—(.RM) [position]

Issues a .BREAK command and sets the position of the right margin. Specify *position* as one of the following:

- Unsigned integer—The exact position of the right margin.
- Integer preceded by a plus sign—A value to be added to the current position of the right margin.
- Integer preceded by a minus sign—A value to be subtracted from the current position of the right margin.

Position defaults to 70.

Initial default: .RIGHT MARGIN 70

.SAVE—(.SA)

Saves the current formatting context which includes the following settings: date status, fill, flags, headers, justification, keep status, margins, numbering, page size, paging, paragraph parameters, spacing, subtitles, and tab stops. The .SAVE command must be paired with a .RESTORE command. You can nest one saved-formatting context within another to a depth of 20.

.SEND TOC—(.STC) toc-line

Writes a line to the table of contents. The line can contain text, DSR commands, and DSR flags.

.SET DATE—(.SDT) [[day],[month],[year]]

Issues a .BREAK command and sets the date. Specify *day, month,* or *year* as follows:

- Unsigned integer—The day of the month, month, or year. You can specify the entire year or the last two digits.
- Integer preceded by a plus sign—A value to be added to the current value of *day, month,* or *year.*
- Integer preceded by a minus sign—A value to be subtracted from the current value of *day, month,* or *year.*

You can omit a parameter by including only the comma. Omitted parameters default to the current values.

If you omit the parameters entirely, the date is reset to the current date (the date on which the DSR processing occurs).

Initial default: .SET DATE current-date

.SET LEVEL—(.SL) [level]

Sets the level used in .HEADER LEVEL commands. Specify level as one of the following:

- Unsigned integer in the range 1 to 6—The exact level of the header.
- Integer preceded by a plus sign—A value to be added to the current level setting.
- Integer preceded by a minus sign—A value to be subtracted from the current level setting.

Level defaults to the last level specified in a .HEADER LEVEL or .SET LEVEL command (or 1 if a level has not yet been set).

.SET PARAGRAPH—(.SPR) [spaces],[skip-lines],[test-lines]

Sets the parameter values used in .PARAGRAPH commands. Specify *spaces* as one of the following:

- Zero or unsigned integer—The number of spaces to indent the first line to the right of the left margin.
- Integer preceded by a minus sign—The number of spaces to indent to the left of the left margin. You cannot indent past position 0.

Specify *skip-lines* as one of the following:

- Zero or unsigned integer—The number of blank lines to be inserted before the paragraph.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Specify *test-lines* as a factor in determining the number of lines that must be left on the page for the paragraph to start on the page. Otherwise, the paragraph starts on the next page. The precise algorithm is as follows, where *spacing* is the current value of the .SPACING command (initial default is 1).

```
(skip-lines + test-lines + 1 ) * spacing
```

The parameters default to the values specified in the last .PARAGRAPH or .SET PARAGRAPH command.

.SET TIME—(.STM) [[hour],[minute],[second]]

Issues a .BREAK command and sets the time. Specify *hour, minute,* or *second* as follows:

- Unsigned integer—The hour of the day, minute of the hour, or second of the minute.
- Integer preceded by a plus sign—A value to be added to the current hour, minute, or second.
- Integer preceded by a minus sign—A value to be subtracted from the current hour, minute, or second.

You can omit parameters by including only the comma. Omitted parameters default to the current values.

If you omit the parameters entirely, the time is reset to the current time (the time at which the DSR processing occurs).

Initial default: .SET TIME current-time

.SKIP—(.S) [lines]

Issues a .BREAK command and inserts blank lines. *Lines* must be one of the following. The value is multiplied by the current value of the .SPACING command (initial default is 1).

- Zero or unsigned integer—The number of blank lines to be inserted.
- Integer preceded by a minus sign—The number of lines from the bottom of the page where writing will resume.

Lines defaults to a value of 1.

The .SKIP command does not work at the top of a page unless *lines* is negative. (Use the .FIGURE command.) The .SKIP command does not continue to the next page if *lines* is greater than the number of lines left on the page. If the page contains a footnote, the line directly above the footnote is considered the bottom of the page.

.SPACING—(.SP) lines

Sets the spacing between lines of text, where 1 means single spacing (no blank lines between lines of text), 2 means double spacing, and so on. The following commands multiply a *lines* specification by the current .SPACING value to determine the number of physical lines: .AUTOPARAGRAPH, .AUTOTABLE, .PARAGRAPH, .SET PARAGRAPH, and .SKIP.

Specify *lines* as an integer in the range 1 through 5.

Initial default: .SPACING 1

.STYLE HEADERS—(.STHL) [code],...

Issues a .BREAK command and sets the formatting for level headers. You can specify up to nine codes as follows. The codes must be written in the specified position. Specify a default value by including just the comma.

1. Run-in format—An integer in the range 0 through 7 specifying the lowest numbered level of header to have a run-in format. (A run-in format starts the text on the same line as the header.) Defaults to 3.

2. Uppercase title—An integer in the range 0 through 7 specifying the highest numbered level of header to have its title written in all uppercase letters. Overrides any conflicting code 3 setting. Defaults to 1.

3. Initial capitals in title—An integer in the range 0 through 7 specifying the highest numbered level of header to have its title written in uppercase and lowercase (that is, initial capitals). Defaults to 6.

4. Section number—An integer in the range 0 through 7 specifying the lowest numbered level of header not to have a section number written to the left of the title. Defaults to 7.

5. Centered title—An integer in the range 0 through 7 specifying the lowest numbered level of header to have its header centered. (However, run-in headers are not centered.) Defaults to 7.

6. Preceding blank lines—An unsigned integer specifying the number of blank lines preceding a header. Defaults to 2.

7. Following blank lines—An unsigned integer specifying the number of blank lines following a header. Defaults to 1.

8. Test page lines—An unsigned integer specifying the number of lines that must be left on a page for the level header to be written to that page. Otherwise, the header is written at the top of the next page. Defaults to 7 plus the most recent value specified for the test-lines parameter of a .PARAGRAPH or .SET PARAGRAPH command.

9. Spaces after section number—An integer in the range 1 through 75 specifying the number of spaces between the header number and title. Defaults to 2.

Initial default: .STYLE HEADERS 3,1,6,7,7,2,1,9,2

.SUBPAGE—(.SPG)

Issues a .BREAK command and starts a new page. Initiates subpage numbering: each page has the number of the most recent page immediately followed by a subpage appendix. For example, page 1-12 might be followed by subpages 1-12A and 1-12B. Must be paired with an .END SUBPAGE command.

.SUBTITLE—(.ST) [subtitle]

Issues a .BREAK command and sets *subtitle* as the running head to be used. *Subtitle* must be specified as a character string. A .SUBTITLE command must be the last command on a line.

.TAB STOPS—(.TS) [position],...

Sets the tab stops. Specify *position* as one of the following:

- Unsigned integer—The exact position of the tab stop.
- Integer preceded by a plus sign—A value to be added to the current position of the tab stop.
- Integer preceded by a minus sign—A value to be subtracted from the current position of the tab stop.

Tab stops are set left to right as specified. Each tab stop must be at least two greater than the preceding tab stop. You can retain the value of the previous tab stop by specifying just a comma; you must specify the commas to retain trailing tab stops.

Initial default: .TAB STOPS 8,16,24,...

.TEST PAGE—(.TP) lines

Issues a .BREAK command and checks the current page for the specified number of remaining lines. If the page does not contain enough room for the specified number of lines, a new page is started. *Lines* must be specified as an unsigned integer.

.TITLE—(.T) [title]

Issues a .BREAK command and sets *title* as the running head title to be used. *Title* must be specified as a character string. A .TITLE command must be the last command on a line.

.VARIABLE—(.VR) name [true, false]

Identifies text and commands contained in conditional blocks if the /DEBUG qualifier is specified with the RUNOFF command.

Name must be the name of a variable in an .IF or .IFNOT command. The .IF or .IFNOT command must follow the .VARIABLE command.

True must be a single character. This character (followed by a space) will appear as the first character of each line which is true for the specified variable.

False must be a single character. This character (followed by a space) will appear as the first character of each line which is false for the specified variable.

.XLOWER—(.XL)

Uppercases and lowercases index entries exactly as they appear in the DSR file.

.XUPPER—(.XU)

Uppercases the first character of an index entry and lowercases the remaining characters, except where explicitly overridden by the capitalize or uppercase flag.

Initial default.

1 DSR Flags

The following conditions must be met for a flag to be used:

- Master recognition—The .FLAGS ALL command must be in effect, except for the comment and control flags.
- Individual recognition—The specific .FLAGS command must be in effect.
- Individual enabling—The specific .ENABLE command must be in effect.

If a character is not recognized as a flag, the character is treated as normal text. If a character is not enabled as a flag, the character does not cause the flag action to occur.

The flags are as follows. The second column lists the initial default character representing each flag.

Flag Name	D	Description
ACCEPT	_	Treats the next character as text even if the character is a flag that is in effect, a period, or a space.
BOLD	*	Writes the next character in bold (by overstriking).
BREAK	|	Permits a word to be broken between lines at the point where the flag appears. No hyphen is automatically inserted.
CAPITALIZE	<	Capitalizes all characters that follow until one of the following occurs: an expandable space, a BREAK flag, a HYPHENATE flag, a CAPITALIZE flag, a pair of UPPERCASE flags, a pair of LOWERCASE flags, the end of the line.
COMMENT	!	Treats the characters that follow as a comment until a semicolon or the end of the line occurs. A comment can only occur where a DSR command is legal; otherwise, the COMMENT flag is treated as text.
CONTROL	.	Treats the characters that follow as a DSR command until a semicolon or the end of the line occurs. The CONTROL flag is treated as text if it occurs where a DSR command is not legal.
HYPHENATE	=	Permits a word to be broken between lines at the point where the flag appears. A hyphen is automatically inserted.
INDEX	>	Treats the characters that follow as an index entry.
LOWERCASE	\	Lowercases the next character.
OVERSTRIKE	%	Overstrikes the preceding character with the next character.
PERIOD	+	Inserts an expandable space when the flag is placed after the last character in a word.
SPACE	#	Produces one unexpandable space. The word preceding the flag, the flag, and the next word are all treated as one word in processing.
SUBINDEX	>	In an .INDEX command, treats the characters that follow as a subentry. In an .ENTRY command, treats the characters that follow as a cross-reference.
SUBSTITUTE	$	Must be paired with itself. See the next table.
UNDERLINE	&	Underlines the next character.
UPPERCASE	^	Uppercases the next character.

The following flag pairs can be used. The second column lists the default characters representing each flag pair.

Flag Pair	D	Description
CONTROL COMMENT	.!	Makes an entire line a comment.
CONTROL CONTROL	..	Inserts an actual period in the text.
LOWERCASE BOLD	*	Ends the bolding of characters.
LOWERCASE LOWERCASE	\\	Writes the text that follows in lowercase until another case flag occurs.
LOWERCASE UNDERLINE	\&	Ends the underlining of characters.

Flag Pair	D	Description
SUBSTITUTE SUBSTITUTE	$$	Inserts the date, time, or a part of the date or time depending on a keyword that follows the SUBSTITUTE SUBSTITUTE flag pair. The keyword must immediately follow the flag pair and can be uppercase or lowercase. The valid keywords are as follows: DATE, TIME, YEAR, MONTH, DAY, HOURS, MINUTES, SECONDS.
UNDERLINE SPACE	&#	Inserts an actual underscore into the text.
UPPERCASE BOLD	^*	Bolds the characters that follow until a LOWERCASE BOLD flag pair occurs.
UPPERCASE CAPITALIZE	^ <	Capitalizes the characters that follow until another case flag occurs.
UPPERCASE UNDERLINE	^&	Underlines the characters that follow until a LOWERCASE UNDERLINE flag pair occurs.
UPPERCASE UPPERCASE	^^	Writes the text that follows as it appears in the DSR file until another case flag occurs.

2 Index Formatting

Indexes are formatted according to the following rules:

- Punctuation—A comma separates an index entry or subentry from its page reference. Commas separate multiple page references for the same entry. No comma follows an entry that does not have a page reference.
- Position of subentries—Subentries are positioned under an entry and indented two spaces on their own lines.
- Case—The .XLOWER and .XUPPER commands control whether index entries are uppercase or lowercase.
- Merging—Index entries merge with other entries having identical spelling, spacing, punctuation, and emphasis.

  If .XLOWER is in effect, uppercase characters sort before lowercase characters. If .XUPPER is in effect, uppercase and lowercase entries are merged.

  Entries with different emphasis are sorted in the following order: bolded and underlined, bolded, underlined, no emphasis.
- Sorting .ENTRY entries—Entries without page references are sorted at the beginning of each subindex level.

EDT Editor

The DCL command EDIT invokes the EDT editor. The following section describes keypad and line editing commands available in EDT and commands used to redefine keys in EDT. See the DCL command descriptions in the Reference Section for a description of the DCL command EDIT.

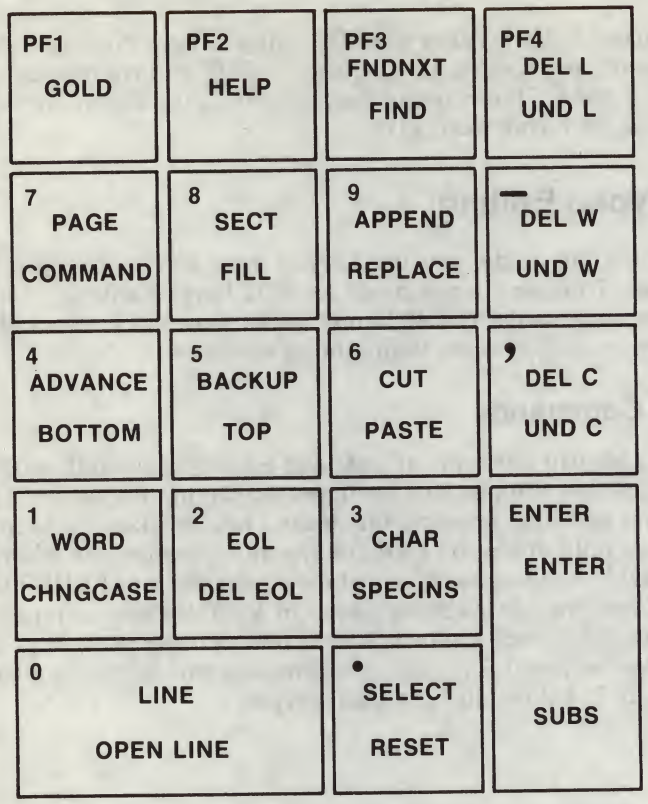
1 EDT Keypad Editing

In EDT keypad-editing mode, you use keypad keys and control keys to perform editing functions. To enter change mode for EDT keypad editing, you must enter the EDT line-editing command CHANGE; text appears on the screen and the keypad-editing keys immediately assume their editing functions.

1.1 Keypad Commands

Each key on the keypad performs at least one editing command; most perform two. Pressing a key invokes the primary (or upper) function. Pressing the GOLD key (labeled PF1) first and then pressing the desired key invokes the alternate (or lower) function. (Do not hold down the GOLD key while pressing the other editing key.) In examples, GOLD key sequences are shown with the word *GOLD* followed by a backslash (\) and the other editing key. On VT200-series terminals, there is a smaller supplemental editing keypad located between the main keyboard and the EDT keypad. See Section 1.3 for more information on the supplemental editing keypad. Figure EDT–1 illustrates the EDT keypad.

Figure EDT-1: EDT Keypad Keys



ZK-1688-84

A description for each key from the diagram follows:

- ADVANCE—Sets the cursor direction forward, from top to bottom of the buffer, for the following commands: CHAR, WORD, FIND, FNDNXT, CHNGCASE, LINE, EOL, PAGE, SECT, and SUBS, and remains in effect until after you press BACKUP.
- APPEND—Deletes the select range (see SELECT) and stores it at the end of the PASTE buffer without otherwise modifying the PASTE buffer's original contents.
- BACKUP—Sets cursor direction backward, from bottom to top of buffer, for the following commands: CHAR, WORD, FIND, FNDNXT, CHNGCASE, LINE, EOL, PAGE, SECT, and SUBS, and remains in effect until you press ADVANCE.
- BOTTOM—Moves the cursor to the end, or bottom, of the buffer.

- CHAR—Moves the cursor one character in the current direction (depending upon whether ADVANCE or BACKUP is set).
- CHNGCASE —Changes the case (from uppercase to lowercase, or lowercase to uppercase) of all letters in the selected range (see SELECT) or search string (see SET SEARCH). If there is no selected range or the cursor is not positioned on the search string, CHNGCASE changes the case of the current character.
- COMMAND—Invokes the *Command:* prompt for entering a line-editing command. Use the ENTER key to process a line-editing command issued at the *Command:* prompt.
- CUT—Deletes the selected range (see SELECT) from the current buffer and places the text in the PASTE buffer.
- DEL C—Deletes the character on which the cursor is positioned.
- DEL EOL—Deletes the text from the cursor to the end of the current line, excluding the line terminator. If the cursor is already at the end of a line, DEL EOL deletes the next line.
- DEL L—Deletes text from the cursor to the end of the line, including the line terminator. If the cursor is at the beginning of the line, the entire line is deleted, positioning the cursor at the beginning of the next line.
- DEL W—Deletes the text from the cursor to the first character of the next word. The line terminator (EOL) is treated as a word by the DEL W command.
- ENTER—Enters the response to a *Search for:* or *Command:* prompt, or completes the processing of the key definition operation.
- EOL—Moves the cursor to the end of the current line or the previous line (depending upon whether ADVANCE or BACKUP is set). If the cursor is already at the end of the line, EOL moves it to the end of the next or previous line.
- FILL—Formats a select range of text by filling each line with as many whole words as possible within the defined line width. (The SET WRAP command defines line width.)
- FIND—Elicits the *Search for:* prompt as the first step in the FIND operation. The command sequence is: FIND (type the search string after the prompt) and ENTER (or ADVANCE or BACKUP). If the string is found, the cursor moves to the first character in the string; otherwise, the cursor remains in place and the message "String was not found" appears. The default search string is the line terminator (EOL).
- FNDNXT—Moves the cursor to the first character of the next occurrence of the search string specified in the FIND command. If there is no further occurrence of the string, the cursor remains in place and the message "String was not found" appears.

- GOLD—When pressed before another keypad key, specifies that key's alternate function. When pressed before a number and another keypad command, GOLD causes the command to be performed the number of times specified by the number. When used with SPECINS, inserts a character from the DEC Multinational Character set (see SPECINS command). GOLD can be used to define GOLD/keyboard key sequences (see CTRL/K).
- HELP—Displays a diagram of keypad keys. When one of the keys is pressed after HELP, information about that key is displayed. This function has no effect on the text you are editing. Press the spacebar to return to keypad editing.
- LINE—Moves the cursor to the beginning of the next or previous line (depending upon whether ADVANCE or BACKUP is set).
- OPEN LINE—Inserts a line terminator at the current cursor position.
- PAGE—Moves the cursor to the next or previous page boundary (depending upon whether ADVANCE or BACKUP is set). The page entity defaults to the text between form feeds and can be defined with the SET ENTITY command.
- PASTE—Inserts the contents of the PASTE buffer (the text last affected by the CUT or APPEND command) at the cursor's current position, positioning the cursor at the end of the inserted text.
- REPLACE—Deletes the selected range and replaces it with the contents of the PASTE buffer (the text last affected by the CUT or APPEND command). The command sequence is store the new text in the PASTE buffer with SELECT and CUT; locate the old text with FIND and ENTER; mark a selected range of the text to be replaced; and press REPLACE. If a range of text to be replaced is not selected, the text of the search string will be replaced.
- RESET—Cancels a selected range, and sets EDT's current direction to ADVANCE.
- SECT—Moves the cursor 16 lines in the current direction (depending upon whether ADVANCE or BACKUP is set).
- SELECT—Marks the current cursor position as the beginning of a selected range. The selected range consists of all the text between this marked position and the cursor's subsequent position at the other end of the selected text.
- SPECINS—Inserts a character (using its decimal value) from the DEC Multinational Character set (see Appendix A). The command sequence is GOLD, the decimal value of the character, GOLD, and SPECINS.
- SUBS—Deletes the selected range, replaces it with the contents of the PASTE buffer, and moves the cursor (in the direction set by ADVANCE or BACKUP) to the next occurrence of the search string. The command sequence is store the new text in the PASTE buffer with SELECT and CUT; locate the old text with FIND and ENTER; replace the old text with the new text and find the next occurrence of the search string using SUBS. If the string is not found, the cursor remains in

place and the message "String was not found" appears. If the string is found, you can replace any subsequent occurrences of the search string by entering SUBS.

- TOP—Moves the cursor to the beginning, or top, of the buffer.
- UND C—Inserts at the cursor's current position the character most recently deleted with DEL C or DELETE.
- UND L—Inserts at the cursor's current position the line of text most recently deleted with DEL L, DEL EOL, or CTRL/U.
- UND W—Inserts at the cursor's current position the word most recently deleted with DEL W or LINEFEED.
- WORD—Moves the cursor one WORD in the current direction (depending upon whether ADVANCE or BACKUP is set). You can define the WORD entity with the SET ENTITY command.

You can also use the arrow keys to move the cursor.

1.2 Keyboard Keys

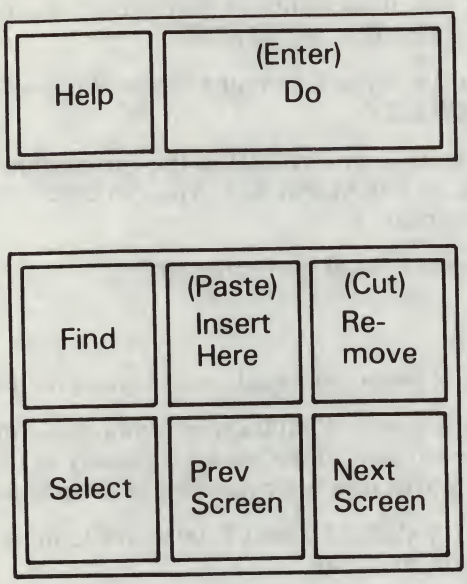
You can use the following keyboard keys to supplement the keypad:

- F12 (the BACKSPACE key on VT100-series terminals)—Moves the cursor to the beginning of the current line. If the cursor is already at the beginning of a line, F12 (BACKSPACE) moves it to the beginning of the previous line.
- The Delete (<X) key (DELETE on VT100-series terminals)—Deletes the character to the left of the cursor.
- F13 (the LINEFEED key on VT100-series terminals)—Deletes the text from the cursor back to the beginning of the word. If the cursor is on the first character in a word, F13 (LINEFEED) deletes the previous word.
- RETURN—Inserts a line terminator at the current cursor position.
- TAB—Moves the text to the next tab stop.

1.3 VT300 and VT200 Supplemental Editing Keypad

The VT300- and VT200-series terminals have a supplemental editing keypad, which is illustrated in the Figure EDT–2.

Figure EDT–2: EDT Supplemental Keypad Keys



ZK-1677-84

The VT300 and VT200 supplemental editing keypad keys perform the same functions as some of the EDT keypad keys, and are described below:

- DO—Enters the response to a *Search for:* or *Command:* prompt or completes the processing of line-editing commands. (Performs the same function as the ENTER keypad key.)
- FIND—Elicits the *Search for:* prompt as the first step in the FIND operation. Type the search string after the prompt and then press either the DO or ENTER key to process the search. (Performs the same function as the FIND keypad key.)
- HELP—Displays a diagram of EDT keypad keys. When one of the keys is pressed after HELP, information about that key is displayed. This function has no effect on the text you are editing. (Performs the same function as the HELP keypad key.)
- INSERT HERE—Inserts the contents of the PASTE buffer at the cursor's current position. (Performs the same function as the PASTE keypad key.)

- NEXT SCREEN—Moves the cursor forward 16 lines. (Performs the same function as the ADVANCE SECT keypad sequence.)
- PREV SCREEN—Moves the cursor backward 16 lines. (Performs the same function as the BACKUP SECT keypad sequence.)
- REMOVE—Deletes the select range (see SELECT) from the current buffer and places the text in the PASTE buffer. (Performs the same function as the CUT keypad key.)
- SELECT—Marks the current cursor position as the beginning of the select range. The select range consists of all the text between this marked position and the cursor's subsequent position at the end of the text being selected. (Performs the same function as the SELECT keypad key.)

1.4 Control Keys

You can use the following control keys in EDT keypad editing.

- CTRL/A—Establishes the current cursor position and resets the indentation level count to be the quotient of the cursor position divided by the SET TAB value. The cursor position must be a multiple of the SET TAB value (a valid tab stop); otherwise, EDT returns the message "Could not align tabs with cursor" and no action is taken. (GOLD + A also performs this function.)
- CTRL/C—Aborts the currently executing EDT command.
- CTRL/D—Decreases the TAB indentation level count one tab setting. (GOLD + D also performs this function.)
- CTRL/E—Increments the TAB indentation level count by one. (GOLD + E also performs this function.)
- CTRL/H—Like BACKSPACE, moves the cursor to the beginning of the line (or to the beginning of the preceding line if the cursor is already at the beginning of the line).
- CTRL/I—Like TAB, moves a line of text to the next tab stop.
- CTRL/J—Like LINEFEED, deletes backward from the cursor to the beginning of a word. If the cursor is on the first character in a word, CTRL/J deletes the previous word.
- CTRL/K—Redefines keypad, control keypad, and gold keypad keys for the current terminal session. When you press CTRL/K, the message "Press the key you wish to define" appears. After you have responded, the message "Now enter the definition terminated by ENTER" appears. Enter the key definition, either by pressing one or more EDT keypad keys or by typing EDT nokeypad editing commands (or a combination of pressing keypad keys and typing nokeypad commands). The guidelines for valid definitions are identical to the DEFINE KEY command, except that you do not need to include the delimiting quotation marks. Terminate the definition by typing a period and pressing the ENTER key.

The following key definition redefines CHAR to transpose the two characters to the left of the cursor. It moves the cursor two characters to the left (← ←); deletes the character at the new cursor position (DEL C); moves the cursor one character to the right (→); restores the deleted character (GOLD and UND C); and then moves the cursor back to its previous position (→).

CTRL/K

```
Press the key you wish to define:

Now enter the definition terminated by ENTER
```

To complete the key definition, type a period on the main keyboard and then press the ENTER key.

- CTRL/L—Inserts a form feed character.
- CTRL/M—Like RETURN, inserts a carriage return into your text.
- CTRL/R—Clears and refreshes the screen, removing extraneous characters and restoring the previous display.
- CTRL/T—If you have previously specified a SET TAB value, CTRL/T indents whole lines in the select range one tab stop to the right. (GOLD + T also performs this function.)

  NOTE: To allow CTRL/T to work correctly in EDT, you must first disable the DCL function of CTRL/T. By default, DCL displays process statistics when you enter CTRL/T. To disable the DCL function, enter the DCL command SET NOCONTROL=T at the dollar-sign prompt.

- CTRL/U—Deletes from the cursor to the beginning of the line. If the cursor is positioned at the beginning of the line, CTRL/U deletes the previous line.
- CTRL/W—Clears and refreshes the screen, removing extraneous characters and restoring the previous display.
- CTRL/Z—Changes editing mode from keypad editing to line editing. (GOLD + Z also performs this function.)

EVE Commands

This section describes each EVE command. The commands are listed in alphabetical order and include a list of any default key definitions. If you are a new user of EVE, read Chapter 8 in this manual. If you are already familiar with previous versions of EVE, you should read the EVE help topic called NEW FEATURES to become familiar with the new and enhanced commands of this version.

@

format

@ *init-filespec*

parameter

init-filespec
The initialization file you want to execute. The default file type is EVE. You can use logical names in the file specification, but not wildcards. You cannot nest initialization files.

description

Executes an EVE initialization file during your editing session. An initialization file contains EVE commands, typically to set editing values (such as margins or tabs) or to execute a series of related commands at the same time. Each command in the initialization file must be on a separate line (with no continuations). If a command is incomplete or ambiguous, EVE prompts you the same as if you typed the command interactively. Comments must begin with an exclamation point and must be on lines separate from commands. For more information on using EVE initialization files, see the EVE help topic called INITIALIZATION FILES.

example

The following example executes an initialization file named MYEVE.EVE:

```
Command: @SYS$LOGIN:MYEVE
```

ATTACH

format

ATTACH *[process-name]*

parameter

process-name
The name of the process or subprocess to which you want the terminal to attach. Process names are case-sensitive and must be from 1 to 15 alphanumeric characters. You cannot specify a numeric process ID. If you do not specify a process, you attach to the parent process.

description

Suspends the current EVE session, without ending it, and connects your terminal to another process or subprocess. The process or subprocess must already exist; ATTACH does not create it. (To find out the names of your processes and subprocesses, use the the DCL command SHOW PROCESS /SUBPROCESS.)

example

In the following example, the DCL command SPAWN creates subprocess Smith_1, invoking EVE to edit a file name MEMO.TXT. The EVE command ATTACH then returns control to the parent process (Smith). After completing work in the parent process, the DCL command ATTACH Smith_1 resumes the editing session.

```
$ SPAWN EDIT/TPU memo.txt
%DCL-S-SPAWNED, process Smith_1 spawned
%DCL-S-ATTACHED, terminal now attached to process Smith_1
      .
      .  [ in MEMO.TXT buffer (subprocess Smith_1) ]
      .
Command: ATTACH
%DCL-S-RETURNED,control returned to parent process Smith
 $
      .
      .  [ at DCL level (process Smith) ]
      .
 $ ATTACH Smith_1
      .
      .  [ in MEMO.TXT buffer (subprocess Smith_1) ]
      .
```

BOTTOM

format

BOTTOM

VT300, VT200:

GOLD/↓

VT100:

GOLD/↓

description

Moves the cursor to the end of the current buffer (unless you are already there). The bottom of the buffer is marked [End of File].

BUFFER

format

BUFFER *buffer-name*

parameter

buffer-name

Optionally, the name of the buffer you want to edit or create. You can abbreviate the buffer name. If more than one name matches your request, EVE shows a list of the matching names and recalls the BUFFER command so you can choose the one you want. You cannot use wildcards in the buffer name. (For example, an asterisk is treated as a character in the buffer name.)

description

Puts a buffer in the current window and moves the cursor to your last position in that buffer. If the buffer does not exist, EVE creates a new buffer and puts the cursor at the top of that buffer. You can also use the BUFFER command to view system buffers (created by the editor) such as the message buffer and the $DEFAULTS$ buffer.

To return to a buffer you previously viewed, use the BUFFER command again and specify the name of that buffer. Typically, a buffer has the same name as the file it contains. To find out the names of your buffers, use the SHOW BUFFERS command.

example

The following command puts the cursor in a buffer named TEST, creating the buffer if it does not already exist:

```
Command: BUFFER TEST
```

CAPITALIZE WORD

format

CAPITALIZE WORD

description

Capitalizes the current word or the text highlighted by FIND or SELECT, or WILDCARD FIND, making the first letter uppercase and the remaining letters lowercase. If you are between words, the next word is capitalized. There is no effect on nonalphabetic characters.

CENTER LINE

format

CENTER LINE

description

Centers the current line between the left and right margins of the buffer. The cursor can be anywhere on the line of text you want to center. It moves with the line (that is, stays on the same character as the line moves). CENTER LINE removes spaces and tabs at the start and end of the line, and adds spaces at the start of the line.

CHANGE DIRECTION

format

CHANGE DIRECTION

VT300, VT200:

F11

VT100:

PF3

description

Changes the direction of the current buffer from forward to reverse or back. The default direction is forward (toward the left and down). The current direction of the buffer is shown in the status line. The direction of the buffer affects commands such as FIND and MOVE BY LINE and some EDT and WPS keypad functions.

CHANGE MODE

format

CHANGE MODE

VT300, VT200:

F14
Ctrl/A

VT100:

ENTER
CTRL/A

description

Changes the mode of the current buffer from insert to overstrike or vice versa. For editing text, the default mode is insert. For typing or editing command lines, the mode matches your terminal setting (according to the DCL command SET TERMINAL). The current mode of the buffer is shown in the status line.

DCL

format

DCL *dcl-command*

parameter

dcl-command
The DCL command you want to execute. The command must be complete, including all required parameters. If you do not specify a command, EVE prompts for one. Pressing RETURN at the prompt cancels the operation.

description

Executes a DCL command from within your editing session. EVE spawns a subprocess for the DCL command you specify and creates a buffer named DCL to contain the output from the command. EVE splits the window to show the DCL buffer. You can edit the DCL buffer—for example, to copy output from the DCL command (such as the directory list) into another buffer. To remove the DCL window, typically, you use the EVE command ONE WINDOW.

example

The following command splits the screen and displays the DCL command DIRECTORY and its output (the directory listing) in the second window. The cursor remains in the first window.

```
Command: DCL DIRECTORY *.TXT
```

DEFINE KEY

format

DEFINE KEY *[=key-name] eve-command*

parameters

key-name

The name of the key you want to define (see Table EVE–1). Note that the key name must be preceded by an equal sign to distinguish it from the command you are binding to the key. If you do not type a key name on the command line, EVE prompts for one; pressing RETURN, which cannot be redefined, cancels the operation.

eve-command

The command you want to bind to the key, or the name of a keypad function you want to bind to the key (such as an EDT or WPS key). If you do not specify a command, EVE prompts you for one. Pressing RETURN at the prompt cancels the operation.

description

Defines a key to execute a single EVE command or an EDT or WPS keypad function. You can type the key name on the command line or let EVE prompt you to press the key you want to define. (Table EVE–1 lists the key names you can use.)

To show a key definition, use SHOW KEY. To remove a key definition, use UNDEFINE KEY. Key definitions remain in effect throughout the editing session (or until you redefine the key). To save key definitions for future sessions, put DEFINE KEY commands in an initialization file or use the command SAVE EXTENDED EVE to create a section file.

EVE does not let you define the RETURN key (CTRL/M), function keys F1 through F6 (VT300 or VT200), or any typing keys on the main keyboard, including the space bar. However, you can define the combination of the GOLD key and a typing key. You cannot redefine the Do key (VT300 or VT200) or the PF4 key (VT100) unless you have already bound the DO command to another key. (In other words, there must be at least one key defined as DO.) DIGITAL also recommends that you do not define the following keys:

<X or DELETE	Help (VT300 or VT200) or PF2 (VT100)	CTRL/C
CTRL/O	CTRL/Q	CTRL/R (which EVE defines as REMEMBER, to end a learn sequence)
CTRL/S	CTRL/T	CTRL/U (which EVE defines as ERASE START OF LINE)
CTRL/X	CTRL/Y	

For more information, see EVE help on TPU NONDEFINABLE KEYS.

Key definitions done with DEFINE KEY override those done by SET KEYPAD commands. For example, if you define PF4 and then enable the EDT keypad, which usually defines this key, your definition overrides the EDT definition.

Table EVE-1: EVE Key Names

Key Name	VT200 or VT300 Key	VT100 Key
PF1 . . . PF4	PF1 . . . PF4	PF1 . . . PF4
KP0 . . . KP9	0 . . . 9	0 . . . 9
PERIOD	.	.
COMMA	,	,
MINUS	-	-
ENTER	Enter	ENTER
UP	↑	↑
DOWN	↓	↓
LEFT	←	←
RIGHT	→	→
E1	Find	
E2	Insert Here	
E3	Remove	
E4	Select	
E5	Prev Screen	
E6	Next Screen	
HELP or F15	Help	
DO or F16	Do	
BS or BS_KEY		BACK SPACE
DELETE or DEL_KEY	<X DELETE	
LS or LF_KEY		LINE FEED
RETURN or RET_KEY	Return	RETURN
F7 . . . F20		
CTRL/A	Ctrl/A	CTRL/A
.	.	.
.	.	.
.	.	.
CTRL/Z	Ctrl/Z	CTRL/Z

example

The following command defines the F20 key as SHOW BUFFERS:

```
Command: DEFINE KEY SHOW BUFFERS
Press the key that you want to define: F20
Key defined.
```

DELETE

format

DELETE

VT300, VT200:

<X

VT100:

DELETE

description

Deletes the character left of the cursor. In insert mode, the rest of the line moves left one character to close the space. In overstrike mode, the deleted character is replaced by a space. At the start of a line, you delete the carriage return for the previous line; the current line moves up. To put back the character you deleted, use RESTORE CHARACTER.

DELETE BUFFER

format

DELETE BUFFER *buffer-name*

parameter

buffer-name

The name of the buffer you want to delete. The buffer name must match exactly—no wildcards or abbreviations. Typically, a buffer has the same name as the file it contains. For a list of the buffers you have created, use the SHOW BUFFERS command.

description

Deletes the buffer you specify. If you delete a buffer that has been modified and is not empty, EVE prompts you to confirm that you want to delete it. The following table shows the possible responses and the effect of each response. You need only type the first letter of the response.

Response	Effect
DELETE_ONLY	Delete the specified buffer.
WRITE_FIRST	Write out the buffer to a file before deleting it.
QUIT	Cancel the operation; do not delete the buffer. (This is the default response; you can simply press RETURN.)

If you delete the buffer you are currently viewing, EVE displays another buffer—usually the first buffer viewed in the editing session. If you delete the only buffer, and that buffer is named anything other than MAIN, EVE creates a new buffer named MAIN. If the only buffer in the session is named MAIN, then EVE does not delete it.

You should not delete system buffers, such as the message buffer or the INSERT HERE buffer. Some system buffers cannot be deleted and are marked as permanent. See the list displayed by the SHOW SYSTEM BUFFERS command.

example

The following example deletes a buffer named JABBER.TXT:

```
Command: DELETE BUFFER JABBER.TXT
```

DELETE WINDOW

format

DELETE WINDOW

description

Deletes the window in which the cursor is located (if you are using more than one window). EVE then enlarges the remaining window (or windows) accordingly. Deleting a window does not delete the buffer that was displayed in the window. For more information about using multiple windows, see the EVE help topic called WINDOWS.

DO

format

DO

VT300, VT200:

Do

VT100:

PF4

description

Enters an EVE command as follows:

1. Press the Do key (VT300 or VT200) or PF4 key (VT100). The cursor moves to the command window at the bottom of the screen.
2. Type the command at the *Command:* prompt. You can abbreviate the command (see EVE help on the topic called ABBREVIATING).
3. Press the RETURN key. The command is then executed or EVE prompts you for more information, such as required parameters.

Pressing DO twice repeats the last command you entered. If you press DO and press RETURN without typing a command, then no command is executed.

There must be a key defined as DO, and you can have more than one. For example, setting EDT keypad defines GOLD/KP7 as DO; setting the WPS keypad defines GOLD/[(left bracket).

END OF LINE

format

END OF LINE

VT300, VT200:

Ctrl/E
GOLD/→

VT100:

CTRL/E

description

Moves the cursor to the end of the current line, unless you are already there. You can also use CTRL/E (or another key defined as END OF LINE) to move to the end of the command line you are typing or have recalled.

ENLARGE WINDOW

format

ENLARGE WINDOW *integer*

parameter

integer
The number of screen lines you want to add to the current window. The maximum size of a window depends on the size and type of terminal screen you are using. The minimum size is one line of text and one line for the status line.

description

Increases the height of the current window (if you are using two or more windows). EVE shrinks the other window (or windows) accordingly. For more information about using multiple windows, see the EVE help topic called WINDOWS.

example

The following example enlarges the current window by five lines:

```
Command: ENLARGE WINDOW 5
```

ERASE CHARACTER

format

ERASE CHARACTER

description

Erases the character that the cursor is on. In insert mode, the rest of the line moves left one character to close up the space. In overstrike mode, the erased character is replaced by a space. At the end of a line, you erase the carriage return for that line; the next line (if any) moves up. To put back the erased character, use RESTORE CHARACTER.

ERASE LINE

format

ERASE LINE

description

Erases text, starting at the cursor location (current character) and continuing to the end of the current line. The next line (if any) moves up. If you are at the end of a line, you erase only the carriage return for that line; the next line (if any) moves up. To put back the erased text, use RESTORE LINE.

ERASE PREVIOUS WORD

format

ERASE PREVIOUS WORD

description

Erases the previous word or the word the cursor is on. If you are between words or on the first character of a word, you erase the previous word. In the middle of a word, you erase all of that word and the trailing spaces (same as ERASE WORD). At the start of a line, you erase only the carriage return for the previous line (if any), and the current line moves up. To put back the erased text, use RESTORE WORD.

ERASE START OF LINE

format

ERASE START OF LINE

VT300, VT200:

Ctrl/U

VT100:

CTRL/U

description

Erases all characters left of the cursor to the start of the current line. The current character is not erased. Text to the right of the cursor shifts to the left. If you are already at the start of a line, nothing is erased. To put back the erased text, use RESTORE LINE. You can also use CTRL/U (or another key defined as ERASE START OF LINE) to discard a command line you are typing or have recalled.

ERASE WORD

format

ERASE WORD

VT300, VT200:

F13
Ctrl/J

VT100:

Keypad COMMA
CTRL/J

description

Erases the word the cursor is on or the next word. If you are on a word, you erase that word and the spaces before the next word. If you are between words, you erase the next word and the trailing spaces. At the end of a line, you erase only the carriage return for that line; the next line (if any) moves up. To put back the erased text, use RESTORE WORD.

EXIT

format

EXIT

VT300, VT200:

F10
Ctrl/Z

VT100:

CTRL/Z

description

Ends the editing session and, typically, produces a new file (or a new version of an existing file). When you exit, EVE writes out the current buffer, unless you have made no edits or unless there are no changes since you previously wrote out the buffer during the session.

If there is no file specification for the buffer—that is, if you invoked EVE or created the buffer without specifying an input file—EVE prompts for one. Pressing RETURN at the prompt discards the buffer and continues exiting.

If you have modified other buffers (as in editing more than one file in the session), EVE asks if you want to write out those buffers. Respond YES or NO. If necessary, EVE prompts for any output file specifications.

If you have not modified any buffers, EXIT is the same as QUIT. No new files or new versions are produced.

EXTEND ALL

format

EXTEND ALL

description

Compiles all procedures in the current buffer. This is the same as the command EXTEND EVE *. To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save a compiled procedure in a section file for future editing sessions, use the SAVE EXTENDED EVE command.

EXTEND EVE

format

EXTEND EVE { *procedure-name* | * }

parameters

procedure-name
The name of a VAXTPU procedure you want to compile. You can abbreviate the procedure name. If more than one name matches your request, EVE shows a list of the matching names and recalls the EXTEND EVE command so you can choose the one you want.

*

Wildcard symbol, telling VAXTPU to compile all the procedures in the buffer (effectively the same as the EXTEND ALL command).

description

Compiles one or more VAXTPU procedures to extend EVE (synonymous with EXTEND TPU). Compiler messages appear in the message window. To read or review all the compiler messages, use the command BUFFER MESSAGES to put the message buffer in the current window.

To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save a compiled procedure in a section file for future editing sessions, use the SAVE EXTENDED EVE command.

example

The following example compiles a procedure named USER_PROC:

```
Command: EXTEND EVE USER_PROC
```

EXTEND THIS

format

EXTEND THIS

description

Compiles the VAXTPU procedure that the cursor is on. This is the same as the EXTEND EVE command, except that you do not type the procedure name. The cursor can be anywhere in the procedure you want to compile. To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save a compiled procedure in a section file for future editing sessions, use the SAVE EXTENDED EVE command.

FILL

format

FILL

description

Fills (reformats) the current paragraph or the text highlighted by FIND, SELECT, or WILDCARD FIND according to the margins of the buffer, so that the maximum number of words fits on a line. Paragraphs are bounded by any of the following:

- Blank lines
- Top or bottom of the buffer
- Page breaks
- DIGITAL Standard Runoff commands

FILL removes tabs and spaces at the start and end of the paragraph or range, but does not affect tabs and spaces in the middle of the range or within the paragraph.

FILL PARAGRAPH

format

FILL PARAGRAPH

description

Fills (reformats) the current paragraph according to the margins of the buffer, so that the maximum number of words fits on a line. The cursor can be anywhere in the paragraph you want to fill. Paragraphs are bounded by any of the following:

- Blank lines
- Top or bottom of the buffer
- Page breaks
- DIGITAL Standard Runoff commands

The cursor moves to the end of the paragraph. FILL PARAGRAPH removes tabs and spaces at the start and end of the paragraph, but does not affect tabs and spaces within the paragraph.

FILL RANGE

format

FILL RANGE

description

Fills (reformats) the text highlighted by FIND, SELECT, or WILDCARD FIND according to the margins of the buffer so that the maximum number of words fits on a line. FILL RANGE removes tabs and spaces at the start and end of the range, but does not affect tabs and spaces in the middle of the range.

FIND

format

FIND *[search-string]*

VT300, VT200:

Find

VT100:

PF1

parameter

search-string

The string of text you want to find. Use all lowercase to find any occurrence of the string. Use uppercase or mixed case to find an exact match. FIND is also sensitive to diacritical marks, such as accents, in the search string.

description

Searches the current buffer for the text string you specify or for one already entered. If the string is found, EVE highlights the text and puts the cursor at the start of the string.

Pressing FIND twice searches for the last string you entered.

You can use direction-setting keys to terminate FIND commands to specify the direction to begin the search. For example, if you terminate the command by pressing F11, the search begins in the direction opposite the current direction of the buffer; if you press a key defined as FORWARD (such as EDT KP4), the search begins in forward direction regardless of the buffer direction. Pressing RETURN begins the search in the current direction of the buffer.

The FIND command first searches the buffer in the direction you specify or the current direction. If the string is not found, EVE searches in the opposite direction and, if the string is then found, asks if you want to go there. Press RETURN if you want to go there, or type No and press RETURN to end the search.

The found string is highlighted, similar to a select range. You can then use REMOVE, STORE TEXT, LOWERCASE WORD, or other commands that work on selected text. Moving off the string or using RESET cancels the highlighting.

By default, FIND treats white space (tabs and spaces) in the search string literally. In other words, if you search for "Mark Twain", EVE finds the string if there is one space between the words and if both words are on a single line. If you use the SET FIND WHITESPACE command, EVE finds strings like "Mark Twain" whether there are one or more spaces or tabs between the words and whether or not the string is split by a line break (for example, "Mark" at the end of a line and "Twain" at the start of the next line).

FORWARD

format

FORWARD

description

Sets the direction of the current buffer to forward (toward the right and down). The current direction of the buffer is shown in the status line. It affects commands like FIND and MOVE BY LINE and some EDT and WPS keypad functions.

GET FILE

format

GET FILE *filespec*

parameter

filespec

The file you want to edit or create. You can use logical names and wildcards in the file specification. If more than one file matches your request, EVE shows a list of the matching files and recalls the GET FILE command so you can choose the one you want. You can edit several files in an editing session, but can get only one file at a time.

description

Puts a file into the current window, creating a new buffer if necessary. This lets you edit another file in the same session. Getting a file for the first time creates a new buffer with the name of that file. (If there is already a buffer by that name, EVE asks for a different buffer name to use.) If the file you specify is one you have already edited during the session, EVE puts the cursor at your last location in the buffer for that file. In effect, this is the same as using the command BUFFER.

example

The following example gets a file named TEST.DAT, putting into the current window:

```
Command: GET FILE TEST.DAT
```

GO TO

format

GO TO *marker-name*

parameter

marker-name

The marker you want to go to, as previously specified with the MARK command. You can abbreviate marker names. If more than one name matches your request, EVE shows a list of the matching names and recalls the GO TO command so you can choose the one you want. The names are not case sensitive.

description

Moves the cursor to the location previously labeled with the MARK command. Using MARK and GO TO commands makes it easier to move through a large buffer or file, or to move between buffers. If the marker is in a buffer other than the one currently in view, EVE puts that buffer into the current window. To find out marker names, use the SHOW command.

example

The following example puts the cursor at the position previously labeled CHAP 1 with the command MARK:

```
Command: MARK CHAP 1
                .
                .
                .
Command: GO TO CHAP 1
```

HELP

format

HELP *[topic-name]*

VT300, VT200:

Help
GOLD/Help

VT100:

PF2

parameter

topic-name
The EVE command or other topic for which you want help. You can abbreviate topic names so long as they are unambiguous. If you do not specify a topic, or if you specify a question mark (?), EVE displays the list of topics.

description

Displays online help about an EVE command or defined key, or information about other topics, including VAXTPU built-in procedures.

For a keypad diagram, pressing the Help key (VT300 or VT200) or the PF2 key (VT100). For a list of defined keys, press GOLD/Help. You can then press the key that you want help on.

To get help on VAXTPU built-in procedures and other topics, use HELP TPU. This displays help on the EVE command TPU and switches you to a help library on VAXTPU topics. You can then type the name of a built-in, such as COPY_TEXT or GET_INFO. To return to help on the editor, type EVE.

To scroll through lengthy topics, press NEXT SCREEN or PREV SCREEN. To exit from help and resume editing, press RETURN.

example

The following example shows information about the CENTER LINE command:

```
Command: HELP CENTER LINE
```

INCLUDE FILE

format

INCLUDE FILE *filespec*

parameter

filespec
The file you want to include. You can use logical names and wildcards in the file specification. If more than one file matches your request, EVE shows a list of the matching files and recalls the INCLUDE FILE command so you can choose the one you want. You can include only one file at a time.

description

Includes (copies) a file into the current buffer, inserting its contents before the current line. The buffer name does not change.

example

The following example includes a file named TEST.DAT in the current buffer:

```
Command: INCLUDE FILE TEST.DAT
```

INSERT HERE

format

INSERT HERE

VT300, VT200:

Insert Here

VT100:

KP9

description

Inserts ("pastes") at the current position the text you previously stored or removed (that is, the contents of the INSERT HERE buffer). The text is always inserted, whether the mode of the buffer is inset or overstrike. The cursor moves to the end of the inserted text. The text does not automatically rewrap.

INSERT MODE

format

INSERT MODE

description

Sets the mode of the current buffer to insert. In insert mode, each character you enter is entered at the current position (marked by the cursor), pushing existing characters to the right. This is the default setting. The current mode of the buffer is shown in the status line.

INSERT PAGE BREAK

format

INSERT PAGE BREAK

VT300, VT200:

Ctrl/L

VT100:

CTRL/L

description

Inserts a form feed (visible as $\underset{F}{E}$) at the start of a line by itself.

- If you are not at the start of a line, EVE first does a RETURN, then inserts the form feed, and moves the cursor to the start of the next line.
- At the start of a line of text, EVE inserts the form feed and does a RETURN, putting the cursor at the start of the next line.
- At the start of a blank line, EVE inserts the form feed and moves the cursor to the start of the next line, without doing a RETURN.

To erase page breaks, use MOVE BY PAGE to put the cursor on a page break, and then use ERASE LINE or a similar EDT or WPS keypad function.

LEARN

format

LEARN

description

Learns a sequence of keystrokes and remembers them as a single key. The sequence can comprise text, commands, or both, including keys already defined. EVE prompts you to enter the keystrokes you want learned. To end or remember the learn sequence, press CTRL/R (defined as REMEMBER).

Key definitions done with LEARN override those done by SET KEYPAD commands. For example, if you use PF4 for a learn sequence and then enable the EDT keypad, which usually defines this key, your definition overrides the EDT definition.

Learn sequences are remembered throughout the editing session or until you redefine the key. To save key definitions for future sessions, use SAVE EXTENDED EVE to create a section file. You cannot put learn sequences in an initialization file.

example

The following example creates a learn sequence that binds the INSERT HERE and FILL PARAGRAPH commands to a single key (F20):

```
Command: LEARN
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
[DO] INSERT HERE [RETURN]
[DO] FILL PARAGRAPH [RETURN]
[CTRL/R]
Press the key you want to use to do what was just learned: [F20]
Key sequence remembered.
```

LINE

format

LINE *integer [procedure-name]*

parameters

integer

The number of the line in the current buffer where you want EVE to move the cursor. If you do not specify a line number, EVE prompts for one. Simply pressing RETURN at the prompt cancels the command.

procedure-name

Optionally, the name of a VAXTPU procedure in the current buffer. This is useful because some compiler messages refer to line numbers in a procedure.

description

Moves the cursor to the start of a line you specify by number—either in the buffer or within a specified procedure in the buffer. To find out the current line number and total number of lines in the buffer, use the WHAT LINE command.

example

```
Command: LINE 12
```

Moves the cursor to the start of line 12 in the current buffer.

LOWERCASE WORD

format

LOWERCASE WORD

description

Makes the current word or the text highlighted by FIND, SELECT, or WILDCARD FIND all lowercase. If you are between words, the next word is made uppercase. There is no effect on nonalphabetic characters.

MARK

format

MARK *marker-name*

parameter

marker-name
One or more characters to mark your location in the buffer. Marker names are not case-sensitive and may contain spaces and tabs. If you specify a marker name that is already used, the previous marker is canceled.

description

Puts an invisible mark at the current position. Later, using the command GO TO and the marker nameiyou can return to the marked location. This is useful for moving through a large buffer or file, or for moving between buffers.

example

The following example marks the current position as CHAP 1 so you can return to it later by using the GO TO command:

```
Command: MARK CHAP 1
         .
         .
         .
Command: GO TO CHAP 1
```

MOVE BY LINE

format

MOVE BY LINE

VT300, VT200:

F12

VT100:

Keypad MINUS

description

Moves the cursor to the start or end of a line. In forward direction, the cursor moves to the end of the current line or if already there, the end of the next line. In reverse direction, the cursor moves to the start of the current line or if already there, the start of the previous line.

MOVE BY PAGE

format

MOVE BY PAGE

description

Moves the cursor to the next page break in the current direction. A page break is a form feed character (usually appearing as F_F). If there is no next or previous page break, then in forward direction, the cursor moves to the end of the buffer, or in reverse direction, to the top of the buffer.

MOVE BY WORD

format

MOVE BY WORD

description

Moves the cursor a word at a time in the current direction. In forward direction, the cursor moves to the start of the next word (if any)—that is, the first nonspace character in the word. In reverse direction, the cursor moves to the start of the current word or if already there, to the start of the previous word (if any).

MOVE DOWN

format

MOVE DOWN

VT300, VT200:

↓

VT100:

↓
KP2

description

Moves the cursor down a line at a time. If the cursor is free, it moves down in the same column on the screen, regardless of whether characters are already there or not. If the cursor is bound, it moves to the corresponding position in the next line (if any), following the shape of your text. For example, from the end of a line longer than the next line, the cursor moves to the end of the next line.

MOVE LEFT

format

MOVE LEFT

VT300, VT200:

←

VT100:

←
KP1

description

Moves the cursor one character to the left. If the cursor is free, you can move it anywhere in the buffer, whether characters are already there or not. If the cursor is bound, then from the start of a line, it moves to the end of the previous line, if there is one.

MOVE RIGHT

format

MOVE RIGHT

VT300, VT200:

→

VT100:

→
KP3

description

Moves the cursor one character to the right. If the cursor is free, you can move it anywhere in the buffer, whether characters are already there or not. If the cursor is bound, then from the end of a line, it moves to the start of the next line, if there is one.

MOVE UP

format

MOVE UP

VT300, VT200:

↑

VT100:

↑
KP5

description

Moves the cursor up a line at a time (unless you are at the top of the buffer). If the cursor is free, it moves up in the same column on the screen, regardless of whether characters are already there or not. If the cursor is bound, it moves to the corresponding position in the previous line (if any), following the shape of your text. For example, from the end of a line longer than the previous line, it moves to the end of the previous line.

NEXT SCREEN

format

NEXT SCREEN

VT300, VT200:

Next Screen

VT100:

KP0

description

Scrolls down to show the next screen's worth of text (if any)—roughly the length of the current window. The cursor stays in the same relative position (or offset) in the line. Repeat the operation to scroll forward through the buffer.

NEXT WINDOW

format

NEXT WINDOW

description

Moves the cursor to the next window (if you are using two or more windows). For example, if you are in the uppermost of three windows, the cursor moves to the middle window. If you are in the bottommost window, the cursor moves to the uppermost window. If you are using two windows, NEXT WINDOW, PREVIOUS WINDOW, and OTHER WINDOW are the same. For more information about using multiple windows, see the EVE help topic called WINDOWS.

ONE WINDOW

format

ONE WINDOW

description

Restores the screen to a single large window when you are using multiple windows. EVE deletes all the windows except the current window. The buffers associated with those windows are not deleted. For more information about using multiple windows, see the EVE help topic called WINDOWS.

OTHER WINDOW

format

OTHER WINDOW

description

Moves the cursor to the next window (if you are using two or more windows). The cursor moves to your last position in that window. For example, if you are in the bottommost of three windows, the cursor moves to the top window. If you are in the uppermost window, the cursor moves to the middle window. If you are using two windows, NEXT WINDOW, PREVIOUS WINDOW, and OTHER WINDOW are the same. For more information about using multiple windows, see the EVE help topic called WINDOWS.

OVERSTRIKE MODE

format

OVERSTRIKE MODE

description

Sets the mode of the current buffer to overstrike. In overstrike mode, each character you type replaces the character at the current position. The current mode of the buffer is shown in the status line.

PREVIOUS SCREEN

format

PREVIOUS SCREEN

VT300, VT200:

Prev Screen

VT100:

KEYPAD PERIOD

description

Scrolls up to show the previous screen's worth of text (if any)—roughly the length of the current window. The cursor stays in the same relative position (or offset) in the line. Repeat the operation to scroll backward through the buffer.

PREVIOUS WINDOW

format

PREVIOUS WINDOW

description

Moves the cursor to the previous window (if you are using two or more windows). The cursor moves to your last position in that window. For example, if you are in the bottommost of three windows, the cursor moves to the middle window. If you are in the uppermost window, the cursor moves to the bottommost window. If you are using two windows, NEXT WINDOW, PREVIOUS WINDOW, and OTHER WINDOW are the same. For more information about using multiple windows, see the EVE help topic called WINDOWS.

QUIT

format

QUIT

description

Ends the editing session without writing out a new file or new version of an existing file—that is, you discard the edits made during the session, except those you have written out by using WRITE FILE commands.

If there are any buffers that have been modified and not already written out, EVE asks you to confirm that you want to quit (to prevent accidentally discarding your edits). Type YES or NO (and press RETURN). If you have not modified any buffers, then EXIT is the same as QUIT. Thus, you can quit by pressing a key defined as EXIT, such as F10 or CTRL/Z.

example

The following example ends the editing session without saving your edits:

```
Command: QUIT
Buffer modifications will not be saved, continue quitting? Y
```

QUOTE

format

QUOTE

VT300, VT200:

Ctrl/V

VT100:

CTRL/V

description

Enters a control code or other character, such as escape or form feed, either in text or in a command. The character is entered in the buffer according to the current mode shown in the status line (insert or overstrike). Some control codes appear as a backward question mark.

You can also quote a control code or other character when you enter a string for the commands such as FIND and REPLACE.

example

The following example enters an escape character:

```
Command: QUOTE
Press the key to be added: CTRL/[
```

RECALL

format

RECALL

VT300, VT200:

Ctrl/B

VT100:

CTRL/B

description

Recalls a previous EVE command, which you can edit (if necessary) and execute again.

Do not type the command RECALL. If you type RECALL, that command itself is recalled. Instead, use CTRL/B or a key you have defined as RECALL.

REFRESH

format

REFRESH

VT300, VT200:

Ctrl/W

VT100:

CTRL/W

description

Refreshes (repaints) the screen display. Any extraneous characters on the screen or any messages in the message window are erased, and the cursor remains in the same location.

REMEMBER

format

REMEMBER

VT300, VT200:

Ctrl/R

VT100:

CTRL/R

description

Ends a learn sequence so that a series of keystrokes can be assigned to one key. EVE then prompts you to press the key you want to define for the learn sequence. Pressing RETURN, which cannot be defined, cancels the definition.

Do not type the REMEMBER command to end a learn sequence. If you type REMEMBER, that command itself is remembered as part of the sequence. Instead, use CTRL/R or a key you have defined as REMEMBER.

REMOVE

format

REMOVE

VT300, VT200:

Remove

VT100:

KP8

description

Removes ("cuts") the current text highlighted by FIND, SELECT, or WILDCARD FIND, which you can insert elsewhere. The removed text replaces (in the INSERT HERE buffer) whatever you previously removed or stored. To insert the removed text elsewhere, use INSERT HERE.

If you are viewing a list of buffers, REMOVE deletes the buffer whose name the cursor is on. This is the same as using the command DELETE BUFFER, except you do not type the name of the buffer.

REPEAT

format

REPEAT *integer*

parameter

integer

The number of times to repeat the operation. If you do not specify a number, EVE prompts for one. Pressing RETURN at the prompt cancels the operation.

description

Repeats the next keystroke or command a specified number of times.

You cannot use one REPEAT command to multiply the effects of another REPEAT command. If you use two REPEAT commands in a row, the second command supersedes the first.

example

The following example repeats the command ERASE CHARACTER five times—that is, you erase the current character and the next four:

```
Command: REPEAT 5
Command: ERASE CHARACTER
```

REPLACE

format

REPLACE { *"old-string" ["new-string"]* | *old-string [new-string]* }

parameters

old-string
The text you want to find and remove. If you do not specify an old string on the command line, EVE prompts for one. Simply pressing RETURN at the prompt cancels the operation.

new-string
The text to replace the old string. If you do not specify a new string on the command line, EVE prompts for one. The new string can be null.

description

Replaces one text string with another. EVE searches for the old string, highlights the found text, and asks for one of the following choices:

Response	Effect
YES	Replace this one and find the next one. (This is the default; you can simply press RETURN.)
NO	Skip this one and find the next one.
ALL	Replace all the occurrences (no further prompting).
LAST	Replace this one and stop here.
QUIT	Skip this one and stop here.

With YES or ALL, if the search covers the buffer more than once, EVE asks if you want to continue (so you can avoid replacing a string again when the old and new strings are similar). When the operation is finished, EVE tells you how many replacements were made.

If you specify both the old string and the new string in lowercase, EVE finds every occurrence of the old string regardless of case and matches the case appropriately for the replacement. For example, if the old string is capitalized, EVE replaces it with a capitalized version of the new string. If you use any uppercase letter in the old string, EVE searches for exact occurrences of the old string and does not match the case in making the replacement. If you use any uppercase letters in the new string, the replacement is exact.

example

The following example replaces all occurrences of the string "least" with the string "fewest":

```
Command: REPLACE least fewest
Replace? Type yes, no, all, last, or quit: ALL
Replaced 8 occurrences.
```

RESET

format

RESET

VT300, VT200:

GOLD/Select

description

Cancels the following and sets the direction to forward:

- A press of the GOLD key or, with the EDT or WPS keypad, a repeat count with a GOLD-number combination
- Current highlighting of text (as a result of FIND, SELECT, or WILDCARD FIND)
- An incomplete command (such as if EVE prompts you for a required parameter)
- Output from SHOW, SHOW DEFAULTS BUFFER, SHOW SUMMARY, or SHOW WILDCARD (returning you to the buffer in which you were working)

RESTORE

format

RESTORE

VT300, VT200:

GOLD/Insert Here

description

Restores (undeletes) what you last erased or deleted with one of the ERASE commands or with a similar EDT or WPS keypad function. Depending on what you last erased, RESTORE is the same as RESTORE LINE, RESTORE WORD, or RESTORE SENTENCE. The restored text is inserted, whether the mode of the buffer is insert or overstrike. The cursor moves to the end of the restored text. The text does not automatically rewrap.

RESTORE CHARACTER

format

RESTORE CHARACTER

description

Restores (undeletes) what you last erased or deleted with DELETE, ERASE CHARACTER, or similar EDT or WPS keypad functions. Unlike other RESTORE commands, RESTORE CHARACTER is mode-sensitive; that is, RESTORE CHARACTER puts back the character in overstrike mode if the buffer is in overstrike mode, or in insert mode if the buffer is in insert mode.

RESTORE LINE

format

RESTORE LINE

description

Restores (undeletes) what you last erased or deleted with ERASE LINE, ERASE START OF LINE, or similar EDT or WPS keypad functions. The restored text is always inserted, whether the mode of the buffer is insert or overstrike. The text does not automatically rewrap.

RESTORE SENTENCE

format

RESTORE SENTENCE

description

Restores (undeletes) what you last erased or deleted with the WPS Delete Beginning Sentence key (GOLD/F13 or GOLD/CTRL/J). The restored text is always inserted, whether the mode of the buffer is insert or overstrike. The restored text does not automatically rewrap.

RESTORE WORD

format

RESTORE WORD

VT300, VT200:

GOLD/F13

description

Restores (undeletes) what you last erased or deleted with ERASE WORD, ERASE PREVIOUS WORD, or similar EDT or WPS keypad functions. The restored text is always inserted, whether the mode of the buffer is insert or overstrike. The text does not automatically rewrap.

RETURN

format

RETURN

VT300, VT200:

Return
Ctrl/M
Enter

VT100:

RETURN
CTRL/M

description

Inserts a carriage return at the current editing position or terminates a typed command (or response to a prompt). When you are editing text, RETURN starts a new line, moving the cursor and any existing text to the right of the cursor down. When you terminate a command or a response to a prompt, the cursor can be anywhere on the command line. Generally, if a command prompts for required information, simply pressing RETURN at the prompt cancels the command (see the EVE help topic called CANCELLING).

REVERSE

format

REVERSE

description

Sets the direction of the buffer to reverse (toward the left or the start of the line and up). The current direction of the buffer is shown on the status line. It affects commands like FIND and MOVE BY LINE and some EDT and WPS keypad functions.

SAVE EXTENDED EVE

format

SAVE EXTENDED EVE

parameter

section-filespec

The section file you want to create. The default file type is TPU$SECTION.

description

Saves your current key definitions and other extensions in a section file for future editing sessions (synonymous with SAVE EXTENDED TPU). For more information, see the EVE help topic called SECTION FILES.

example

The following example creates a section file named MYEVE.TPU$SECTION:

```
Command: SAVE EXTENDED EVE SYS$LOGIN:MYEVE
```

SELECT

format

SELECT

VT300, VT200:

Select

VT100:

KP7

description

Selects text for an editing operation, such as STORE TEXT, REMOVE, or FILL. The selected text is highlighted in reverse video. You can then use the following commands with the selected text:

- CAPITALIZE WORD
- STORE TEXT
- FILL (or FILL RANGE)
- REMOVE
- SPELL

- LOWERCASE WORD
- UPPERCASE WORD
- Some EDT and WPS keypad functions

To cancel the selection, simply press SELECT again or use the RESET command.

If you are viewing a list of buffers (with SHOW BUFFERS or SHOW SYSTEM BUFFERS), using SELECT puts into the main window the buffer whose name the cursor is on. This is effectively the same as the BUFFER command except you do not type the buffer name.

SET CURSOR BOUND

format

SET CURSOR BOUND

description

Sets the cursor to a bound state, following the flow of your text, the same as EDT or WPS. Using a bound cursor, you cannot move into an unused portion of the buffer (or "white space"). For example, with a bound cursor, if you are at the end of a line and press the right arrow key, the cursor moves to the start of the next line (if any). It does not move past the right margin. By contrast, with a free cursor, you can move anywhere in the buffer whether characters are already there or not. The default setting is free cursor. Using SET KEYPAD WPS also sets the cursor to bound.

SET CURSOR FREE

format

SET CURSOR FREE

description

Sets the cursor to a free or unbound state. This is the default setting. With a free cursor, you can put text anywhere in the buffer, whether characters are already there or not. In other words, the cursor is not bound to the flow of your text. For example, with a free cursor, if you are at the end of a line and press the right arrow key, the cursor moves right, past the end of the line, and you can put text there. In contrast, a bound cursor moves to the start of the next line.

SET FIND NOWHITESPACE

format

SET FIND NOWHITESPACE

description

Sets the FIND command to match spaces and tabs in the search string exactly (rather than as "white space") and not to span line breaks. Thus, a string of two or more words is found only if the string is entirely on one line. This is the default setting.

example

In the following example, EVE finds the words "Mark Twain" if there is exactly one space between the words and they are on the same line:

```
Command: SET FIND NOWHITESPACE
Command: FIND Mark Twain
```

SET FIND WHITESPACE

format

SET FIND WHITESPACE

description

Sets the FIND command to treat spaces and tabs the same (that is, as "white space") and to span up to one line break. Thus, a string of two or more words is found even if there is a line break between two words.

example

In the following example, EVE finds the words "Mark Twain" even if there are one or more spaces or tabs between the words or if "Mark" is at the end of a line and "Twain" at the start of the next line:

```
Command: SET FIND WHITESPACE
Command: FIND Mark Twain
```

SET GOLD KEY

format

SET GOLD KEY *[key-name]*

parameter

key-name
The name of the key you want to set as GOLD (see Table EVE–1). If you do not specify a key name on the command line, EVE prompts for one; simply pressing RETURN (which cannot be redefined) cancels the operation. You cannot make a mouse button the GOLD key.

description

Sets the GOLD key for use with other keys (synonymous with SET SHIFT KEY). You can type the key name on the command line or let EVE prompt you to press the key you want to set as GOLD. Using a GOLD key lets you bind two commands to one key. For example, you can bind ERASE LINE to KP5 on the keypad, and RESTORE LINE to the combination of GOLD and KP5.

You can set only one GOLD key at a time. To cancel or undefine the GOLD key, use the command SET NOGOLD KEY. EVE has no default GOLD key. Using SET GOLD KEY, SET SHIFT KEY, SET KEYPAD EDT, or SET KEYPAD WPS automatically defines some GOLD key combinations, unless you have defined the keys otherwise (see Table EVE–2).

Table EVE–2: VT300/VT200 Keys Defined by Setting the GOLD Key

Key Sequence	Definition
GOLD/F13	RESTORE WORD (except with the WPS keypad)
GOLD/Help	HELP KEYS (list)
GOLD/Find	WILDCARD FIND
GOLD/Insert Here	RESTORE
GOLD/Remove	STORE TEXT
GOLD/Select	RESET
GOLD/↑	TOP
GOLD/←	START OF LINE
GOLD/↓	BOTTOM
GOLD/→	END OF LINE

example

The following example sets PF1 as the GOLD key:

```
Command: SET GOLD KEY PF1
```

SET KEYPAD EDT

format

SET KEYPAD EDT

description

Sets the EDT-style keypad, defining the numeric keypad and other keys. If you have already set a GOLD key, your key is used as GOLD; otherwise, PF1 is GOLD. If you have already defined keys that EDT ordinarily defines, such as KP8 or CTRL/U, your definitions override the EDT definitions.

Setting the EDT keypad does not let you enter EDT commands by using the DO key. EDT functions can be executed only by pressing keys. For more information, see the EVE help topics called EDT DIFFERENCES and EDT CONVERSION.

To keep the EDT keypad for future sessions, use SAVE EXTENDED EVE to create a section file or put the command in an initialization file.

For a list of defined keys, see help on KEYS or press GOLD/Help. For a keypad diagram, press HELP.

SET KEYPAD NOEDT

format

SET KEYPAD NOEDT

description

Cancels the EDT keypad. On VT300-series and VT200-series terminals, this sets the keypad to numeric; on VT100-series terminals, it sets the keypad to VT100.

SET KEYPAD NOWPS

format

SET KEYPAD NOWPS

description

Cancels the WPS keypad. On VT300-series and VT200-series terminals, this sets the keypad to numeric; on VT100-series terminals, it sets the keypad to VT100.

SET KEYPAD NUMERIC

format

SET KEYPAD NUMERIC

description

Sets the keypad to the default state, canceling the EDT, WPS, or VT100 keypad setting. This is the default setting on VT300-series and VT200-series terminals. The command cannot be used on VT100-series terminals.

For a keypad diagram, press the HELP key. For a list of all defined keys, see help on KEYS.

SET KEYPAD VT100

format

SET KEYPAD VT100

description

Sets the EVE default VT100 keypad, canceling the EDT, WPS, or numeric keypad setting. This is useful if you are accustomed to using EVE on a VT100-series terminal. For example:

- PF1 is defined as FIND.
- PF2 is defined as HELP KEYPAD (diagram).
- PF3 is defined as CHANGE DIRECTION.
- PF4 is defined as DO.
- No GOLD key is set.

SET KEYPAD WPS

format

SET KEYPAD WPS

description

Sets the WPS-style keypad, defining the numeric keypad and other keys. If you have already set a GOLD key, your key is used as GOLD; otherwise, PF1 is GOLD. If you have already defined keys that WPS ordinarily defines, such as KP0 or GOLD/R, your definitions override the WPS definitions.

Setting the WPS keypad does not fully implement or emulate WPS. For more information, see the EVE help topic called WPS DIFFERENCES.

To keep the WPS keypad for future sessions, use SAVE EXTENDED EVE to create a section file or put the command in an initialization file.

For a list of defined keys, see help on KEYS or press GOLD/Help. For a keypad diagram, press HELP.

SET LEFT MARGIN

format

SET LEFT MARGIN *integer*

parameter

integer
The column at which you want the left margin. The value must be less than the right margin. The default left margin is 1 (leftmost column). To find out the current margins of the buffer, use the SHOW command.

description

Sets the left margin for the current buffer. EVE uses the left margin for FILL, CENTER LINE, and wrapping text. Setting the margins by itself does not rewrap text. To reformat text according to new margins, use FILL commands. Putting SET LEFT MARGIN in an initialization file sets the left margin of the main buffer and a system buffer named $DEFAULTS$. Buffers you create during the session will have the same left margin as the $DEFAULTS$ buffer.

example

The following example sets the left margin to 5:

```
Command: SET LEFT MARGIN 5
```

SET NOGOLD KEY

format

SET NOGOLD KEY

description

Cancels (undefines) the current GOLD key so you can define the key by itself (synonymous with SET NOSHIFT KEY).

SET NOWRAP

format

SET NOWRAP

description

Disables wrapping at the right margin of the buffer. As you type, your text continues past the right margin and possibly out of view until you press RETURN or use FILL commands. This can be useful for writing very long lines, as in multicolumn tables or source programs with progressively indented statements. To enable wrapping, use SET WRAP, which is the default setting.

SET RIGHT MARGIN

format

SET RIGHT MARGIN *integer*

parameter

integer

The column at which you want the right margin. The value must be greater than the left margin. The default right margin is one column less than the width. Typically, the width is 80 columns; the default right margin is then 79. To find out the current margins of the buffer, use the SHOW command.

description

Sets the right margin for the current buffer. EVE uses the right margin for FILL, CENTER LINE, and wrapping text. Setting the margins by itself does not rewrap text. To reformat text according to new margins, use FILL commands. Putting SET RIGHT MARGIN in an initialization file sets the right margin of the main buffer and a system buffer named $DEFAULTS$. Buffers you create during the session will have the same right margin as the $DEFAULTS$ buffer. Using SET WIDTH makes the right margin of the $DEFAULTS$ buffer one column less than the width. For example, the command SET WIDTH 132 makes the right margin of the $DEFAULTS$ 131.

example

The following example sets the right margin to 65:

```
Command: SET RIGHT MARGIN 65
```

SET SCROLL MARGINS

format

SET SCROLL MARGINS *integer1[%] integer2[%]*

parameters

integer1

The number of lines down from the top of a window at which you want scrolling to begin. If you do not specify a value, EVE prompts you for one; simply pressing RETURN at the prompt keeps the current top scroll margin and prompts you for the bottom scroll margin.

integer2

The number of lines up from the bottom of a window at which you want scrolling to begin. If you do not specify a value, EVE prompts you for one; simply pressing RETURN at the prompt keeps the current bottom scroll margin.

%

Optionally, specifies scroll margins as percentages of the window height, rounded to the nearest line. Specifying percentages instead of numbers of lines is useful when you run EVE on a workstation with a variable-size screen.

description

Sets the distances at the top and bottom of the window at which scrolling begins automatically as you move the cursor. You can specify these distances as numbers of lines down from the top and up from the bottom of the window, or as percentages of the window size.

The default scroll margins are 0 0; that is, scrolling does not start until you move to a line off the current window. You may want to put the SET SCROLL MARGINS command in an initialization file. The scroll margin settings apply to all windows. Therefore, if you use multiple windows, you may want to reset the scroll margins for the smaller windows.

example

Sets the scroll margins at two lines from the top and three lines from the bottom of the window.

```
Command: SET SCROLL MARGINS 2 3
```

SET TABS

format

SET TABS { *AT integer1 [integer2 . . .]* | *EVERY integer* | *INSERT* | *INVISIBLE* | *MOVEMENT* | *SPACES* | *VISIBLE* }

parameters

AT integer1 [integer2 . . .]
The column or columns at which you want a tab stop. The values must be in ascending order and separated by spaces.

EVERY integer
The interval at which you want EVE to set tab stops.

INSERT
Specifies that TAB inserts a tab character at the current position, moving the cursor and any existing text to the right. This is the default setting. Existing tabs are not affected.

MOVEMENT

Specifies that TAB moves the cursor to the next tab stop, but not to move existing text nor insert any characters in the buffer. This makes the TAB key a cursor-moving key only. Existing tabs are not affected.

SPACES

Specifies that TAB inserts the appropriate number of spaces instead of a tab character. Existing tabs are not affected.

VISIBLE

Makes tab characters visible. For example, the following commands set the tab mode to spaces and make existing tabs visible, appearing as HT (horizontal tab).

INVISIBLE

Makes tab characters invisible, appearing as blank space. This is the default setting.

description

Sets tabs stops (AT or EVERY) or the tab mode (INSERT, SPACES, or MOVEMENT; VISIBLE or INVISIBLE). The default tab stops are every eight columns. The default tab mode is insert and invisible. You can specify only one keyword per command. Tab stops are buffer-specific. If you change the tab stops during an editing session, EVE displays all text in the buffer using the new tab stops. To show the tab stops set for the current buffer, use the SHOW command. Tab mode is a global setting (in effect, a way of defining the TAB key). If you use terminals or printers that have tab settings different from those you specified, the file does not appear the same as it does during an EVE editing session. Also, SET TABS commands do not affect the hardware tab settings of your terminal.

example

Sets tab stops at columns 9, 16, and 24.

```
Command: SET TABS AT 9 16 24
```

SET WIDTH

format

SET WIDTH *integer*

parameter

integer
The number of columns you want for the width of the window. If you specify a value greater than 80, EVE sets the terminal to 132-column mode. Do not use a width greater than 80 on VT100-series terminals without the advanced video option (AVO). When the terminal is switched from 80-column mode to 132-column mode, or the reverse, the screen is refreshed.

description

Sets the width of the terminal display (that is, the number of columns in the window). This is useful for viewing or editing very long lines. The default width is the same as your terminal setting at DCL (determined by the DCL command SET TERMINAL). Typically, this is 80 columns. If the width is greater than 80, EVE uses 132-column mode.

The SET WIDTH command does not affect how many characters you can put on a line, but only how many characters on a line are visible. The number of characters you can put on a line is controlled by the SET WRAP and SET RIGHT MARGIN commands. Using SET WIDTH in EVE affects only your editing session.

Using the SET WIDTH command makes the right margin of the $DEFAULTS$ buffer one column less than the width. New buffers you create will have the same right margin as the $DEFAULTS$ buffer. (Margins of other buffers are not affected).

example

The following example sets the screen width to 132 columns:

```
Command: SET WIDTH 132
```

SET WILDCARD ULTRIX

format

SET WILDCARD ULTRIX

description

Enables ULTRIX-like wildcards (regular expressions) for WILDCARD FIND. The ULTRIX-like wildcards include the period (.) to match any single character and the dollar sign ($) to match the end of the line. For a list of the ULTRIX-like wildcards, use the SHOW WILDCARDS command. If you want to use ULTRIX-like wildcards all the time, put SET WILDCARD ULTRIX in an initialization file. The default wildcard setting is VMS.

example

The following example enables ULTRIX-like wildcards and then finds strings such as "but" and "robot" at the end of a line:

```
Command: SET WILDCARD ULTRIX
Command: WILDCARD FIND b.t$
```

SET WILDCARD VMS

format

SET WILDCARD VMS

description

Enables VMS-style wildcards for WILDCARD FIND. This is the default setting. VMS wildcards include the asterisk (*) to match any number of characters on a line, the percent sign (%) to match any single character, and the backslash and right angle bracket (\>) to match the end of the line. For a list of the VMS-style wildcards, use the SHOW WILDCARDS command.

example

The following example sets VMS-style wildcards and then finds strings such as "but" and "robot" at the end of a line:

```
Command: SET WILDCARD VMS
Command: WILDCARD FIND b%t\>
```

SET WRAP

format

SET WRAP

description

Enables wrapping at the right margin of the buffer. As you type, EVE starts new lines without your having to press RETURN or use FILL. This is the default setting. To disable wrapping, use SET NOWRAP.

SHIFT LEFT

format

SHIFT LEFT *integer*

parameter

integer
The number of columns you want to shift the window to the left. You cannot shift the window past column 1.

description

Shifts the current window to the left by the number of columns you specify. Using the SHIFT LEFT and SHIFT RIGHT commands lets you view the unseen portions of very wide lines, without changing the width of the window. To find out the existing shift (if any) for the window, use the command SHOW.

The SHIFT LEFT command does not move text or affect how many characters you can put on a line; it affects which portion of a line is visible at a given time. The number of characters you can put on a line is controlled by the SET WRAP and SET RIGHT MARGIN commands.

example

In the following sequence, the first command shifts the window five columns to the right. This lets you see beyond the current screen width. The second command shifts the window five more columns to the right. The third command shifts the window ten columns to the left. After each command, EVE displays a message telling the cumulative shift to the right.

```
Command: SHIFT RIGHT 5
Window now shifted right 5 columns.
Command: SHIFT RIGHT 5
Window now shifted right 10 columns.
Command: SHIFT LEFT 10
Window now shifted right 0 columns.
```

SHIFT RIGHT

format

SHIFT RIGHT *integer*

parameter

integer
The number of columns you want to shift the window to the right.

description

Shifts the current window to the right by the number of columns you specify. Using SHIFT RIGHT and SHIFT LEFT commands lets you view the unseen portions of very wide lines, without changing the width of the window. To find out the existing shift (if any) for the window, use the command SHOW.

The SHIFT RIGHT command does move text and does not affect how many characters you can put on a line; it affects which portion of a line is visible at a given time. The number of characters you can put on a line is controlled by the SET WRAP and SET RIGHT MARGIN commands.

example

The following example shifts the window right 10 columns, letting you see beyond the current screen width:

```
Command: SHIFT RIGHT 10
```

SHOW

format

SHOW

description

Shows the following information about the buffers created during the editing session:

- Name of the buffer you are editing or viewing
- Input and output file specifications of the buffer (if any)
- Whether the buffer has been modified
- Total number of lines in the buffer
- Buffer settings (such as margins and tabs)
- Names of markers in the buffer (if any)

- List of nondefault key maps for the buffer (if any)

If you have created more than one buffer, EVE first shows information about the buffer you are currently editing. To show information about the other buffers, press the DO key (VT300 or VT200) or the PF4 key (VT100). To resume editing, press any other key or use RESET.

SHOW BUFFERS

format

SHOW BUFFERS

description

Lists the buffers you have created and puts the cursor in the list so you can scroll through the list. You can then use SELECT and REMOVE to view or delete a buffer without having to type its name, as follows:

1. Use the SHOW BUFFERS command.
2. Put the cursor on the name of the buffer you want to view or delete.
3. Press SELECT to view that buffer or press REMOVE to delete it.

The effect is the same as using the BUFFER or DELETE BUFFER command, respectively. These definitions of SELECT and REMOVE apply only when you are viewing a list of buffers.

SHOW DEFAULTS BUFFER

format

SHOW DEFAULTS BUFFER

description

Shows information about the $DEFAULTS$ buffer—margins, tab stops, direction, mode, maximum lines, and so on. The settings of the $DEFAULTS$ buffer are used when you create buffers during a session (by using GET FILE or BUFFER commands).

To change the settings of the $DEFAULTS$ buffer, do the following:

1. Use the command BUFFER $DEFAULTS$ to put the $DEFAULTS$ buffer in the main or current window. The buffer itself is empty.
2. Change the settings of the $DEFAULTS$ buffer using the relevant EVE commands, such as SET LEFT MARGIN, SET RIGHT MARGIN, and SET TABS AT or EVERY.

Table EVE–3 lists the default EVE settings. Note that some settings, such as the keypad and tab mode, are global (appying to the editor or to all buffers); other settings are buffer-specific. For more information, see the EVE help topic called DEFAULTS.

Table EVE–3: EVE Default Settings

EVE Setting	Default Value
	Global Settings
Cursor	FREE (Using SET KEYPAD WPS sets the cursor to BOUND.)
Keypad	NUMERIC (for VT300- and VT200-series terminals) or VT100 (for VT100-series terminals)
Gold key	NOGOLD (none set)
Find	NOWHITESPACE (that is, the FIND command matches spaces and tabs in the search string exactly as entered and does not span line breaks)
Tab mode	INSERT and INVISIBLE (that is, TAB inserts a tab character which appears as blank space)
Width	Same as your terminal setting (typically 80 columns)
Wildcards	VMS style
Scroll margins	0 0 (that is, scrolling does not begin automatically until you move past the top or bottom of the window)
	Buffer Settings
Direction	FORWARD (right and down)
Mode	For editing text, INSERT; for entering commands, the mode is the same as your terminal setting (with the DCL command SET TERMINAL)
Left margin	1 (leftmost column)
Right margin	One column less than the width (typically, the right margin is 79)
Tab stops	EVERY 8 columns
Wrapping	WRAP (that is, EVE wraps text at the right margin of the buffer as you type)

The SET WIDTH command sets the right margin of the $DEFAULTS$ buffer to one column less than the width.

SHOW KEY

format

SHOW KEY *[key-name]*

parameter

key-name
The name of the key you want described (see Table EVE–1).

description

Shows the definition of a key. You can type the key name on the command line or let EVE prompt you to press the key you want to show.

example

The following example shows the definition of GOLD/KP8 when you are using the EDT keypad:

```
Command: SHOW KEY
Press the key to describe: GOLD/KP8
GOLD/KP8 is defined as "fill" in the EDT keypad.
```

SHOW SUMMARY

format

SHOW SUMMARY

description

Shows statistics and other information about EVE, as follows:

- Version number of the software
- Current journal file specification (if any)
- Current section file specification
- Total number of buffers (system- and user-created)
- Modules used in the section file
- Other information about the EVE configuration

This information is useful for VAXTPU programming or in case you need to submit a software performance report (SPR).

To scroll through the list, press NEXT SCREEN or PREV SCREEN. To return to the buffer you were editing, press DO or use RESET.

SHOW SYSTEM BUFFERS

format

SHOW SYSTEM BUFFERS

description

Lists the system buffers (buffers created by EVE) and puts the cursor in the list so you can scroll through the list and specify the buffer you want to view, as follows:

1. Use the command SHOW SYSTEM BUFFERS.
2. Put the cursor on a buffer name, such as MESSAGES.
3. Press SELECT to view that buffer—effectively the same as using the BUFFER command, except you do not type the buffer name. This definition of SELECT applies only when you are viewing a list of buffers.

Do not delete system buffers with names beginning with a dollar sign ($), because these buffers are necessary for some commands to work properly.

SHOW WILDCARD

format

SHOW WILDCARD

description

Lists the wildcards you can use with WILDCARD FIND—either VMS or ULTRIX, depending on your setting. You can then scroll the list, if necessary. To return to the buffer you were editing, press DO or use RESET.

To set the type of wildcards, use SET WILDCARD ULTRIX or SET WILDCARD VMS. The default is VMS. For examples of wildcard searches, see help on the WILDCARD FIND command.

SHRINK WINDOW

format

SHRINK WINDOW *integer*

parameter

integer
The number of screen lines you want to subtract from the current window. The maximum size of a window depends on the size and type of terminal screen you are using. The minimum size is one line of text and one line for the status line.

description

Decreases the height of the current window (if you are using two or more windows). EVE enlarges the other window (or windows) accordingly. For more information about using multiple windows, see the EVE help topic called WINDOWS.

example

The following example shrinks the current window by five lines:

```
Command: SHRINK WINDOW 5
```

SPAWN

format

SPAWN *[command-string]*

parameter

command-string
Optionally, a DCL command (such as MAIL) that you want to run as a subprocess. If you do not specify a command string, EVE spawns a subprocess for DCL.

description

Suspends (but does not end) the editing session and connects the terminal to a new VMS subprocess. To resume the editing session, exit from the utility or, if you are at the DCL prompt, use the LOGOUT command.

example

In the following example, the command SPAWN suspends the current EVE session and creates a new subprocess invoking MAIL. Exiting from MAIL terminates the subprocess and resumes the editing session:

```
Command: SPAWN mail
$ MAIL
    .
    .
    .
MAIL> CTRL/Z
```

SPELL

format

SPELL

description

Runs DECspell (if it is installed on your system) to check the currently selected text or the entire buffer. EVE spawns a subprocess to run DECspell and writes out the current buffer or selected range to a temporary file in SYS$SCRATCH. (The name of the temporary file uses the subprocess PID.) When SPELL finishes, EVE replaces the buffer or range with the new version of the temporary file (containing any corrections) and deletes any old versions of the temporary file. You then resume editing.

Do not use CTRL/Y while using SPELL. This deletes lines in the temporary output file, and therefore destroys the selected range or current buffer.

SPLIT_WINDOW

format

SPLIT_WINDOW *[integer]*

parameter

integer

Optionally, the number of windows you want to create. If you do not specify a number, EVE splits the window in two (same as the command TWO WINDOWS). The maximum number and size of windows depend on the size and type of terminal screen you are using. The minimum size is one line of text and one line for the status line.

description

Splits the current window into two or more windows of equal size. This lets you view or edit different buffers or different parts of the same buffer at the same time. The cursor appears in the new lower window. Initially, both windows contain the same buffer. To put a different file or buffer in a window, use GET FILE or BUFFER commands. To move between windows, use OTHER WINDOW, NEXT WINDOW, or PREVIOUS WINDOW commands. For more information about using multiple windows, see the EVE help topic called WINDOWS.

example

The following example splits the current window into three smaller windows of equal size:

```
Command: SPLIT WINDOW 3
```

START OF LINE

format

START OF LINE

VT300, VT200:

Ctrl/H
GOLD/←

VT100:

Ctrl/H
BACKSPACE

description

Moves the cursor to the start of the current line, unless you are already there. You can also use CTRL/H (or another key defined as START OF LINE) to move to the start of a command line you are typing or have recalled.

STORE TEXT

format

STORE TEXT

VT300, VT200:

GOLD/Remove

description

Copies the currently highlighted text, which you can insert elsewhere. (It does not remove the text from its current location.) The copied text replaces (in the INSERT HERE buffer) whatever you previously copied or removed. To insert the copied text elsewhere, use INSERT HERE. To set tab stops and tab mode, use SET TABS. (For example, SET TABS MOVEMENT makes the TAB key a cursor-moving key only, to move to the next tab stop.) The default tab stops are every eight columns; the default tab modes are insert and invisible. (See the description of the SET TABS command.) To find out the tab stops of the current buffer, use the SHOW command.

TOP

format

TOP

VT300, VT200:

GOLD/↑

description

Moves the cursor to the top of the current buffer (the upper left corner), unless you are already there.

TPU

format

TPU *procedure-name*

parameter

procedure-name
The name of a VAXTPU procedure or statement you want to execute.

description

Executes a VAXTPU procedure or statement during your editing session.

example

The following example executes the COPY_TEXT built-in to enter the current date and time (using the FAO built-in):

```
Command: TPU COPY_TEXT (FAO ("!%D",0));
```

TWO WINDOWS

format

TWO WINDOWS

description

Splits the current window into two windows. This lets you view or edit different buffers or different parts of the same buffer at the same time. The cursor appears in the new lower window. Initially, both windows contain the same buffer. To put a different file or buffer in a window, use GET FILE or BUFFER commands. To move between windows, use OTHER WINDOW, NEXT WINDOW, or PREVIOUS WINDOW commands. For more information about using multiple windows, see the EVE help topic called WINDOWS.

UNDEFINE KEY

format

UNDEFINE KEY *[key-name]*

parameter

key-name
The name of the key you want to undefine (see Table EVE–1). If you do not type a key name on the command line, EVE prompts for one; pressing RETURN (which cannot be undefined) cancels the operation.

description

Undefines a key you have defined with the DEFINE KEY or LEARN command or with the DEFINE_KEY built-in procedure. If the key was defined by a SET KEYPAD command (such as SET KEYPAD EDT or SET KEYPAD WPS), its previous definition (if any) is restored. You can type the key name on the command line or let EVE prompt you to press the key you want to undefine.

example

The following example undefines GOLD/KP8:

```
Command: UNDEFINE KEY
Press the key that you want to undefine: GOLD/KP8
```

UPPERCASE WORD

format

UPPERCASE WORD

description

Makes the current word or the text highlighted by FIND, SELECT, or WILDCARD FIND all uppercase. If you are between words, the next word is made uppercase. There is no effect on nonalphabetic characters.

WHAT LINE

format

WHAT LINE

description

Shows the current line number, total number of lines in the buffer, and percentage of the lines in the buffer above the current line. This is useful if you want to know whether to insert a page break or simply to find out how many lines are in the buffer. For example, EVE shows a message such as the following:

```
Command: WHAT LINE
You are on line 35 out of 45 (78%).
```

To move to a particular line by number, use the LINE command.

WILDCARD FIND

format

WILDCARD FIND *search-string*

VT300, VT200:

GOLD/Find

parameter

search-string
The string of characters, including both exact characters and wildcard characters, that you want EVE to find.

description

Searches the current buffer for a pattern of text, using wildcards, and highlights the found text. To change the wildcard setting, use SET WILDCARD VMS (the default setting) or SET WILDCARD ULTRIX. To get a list of the wildcards, use the command SHOW WILDCARDS.

WILDCARD FIND follows the same rules as FIND for direction and for case sensitivity unless overridden by a wildcard. For example, with VMS-style wildcards, \L (for lowercase) or \U (for uppercase) makes the entire search case sensitive. Pressing the FIND key twice searches for the last string you entered.

example

The following example uses VMS-style wildcards to find a dollar amount (with any number of digits before the decimal point) at the beginning of any line. Thus, EVE finds a string such as "$12.00" at the start of a line.

```
Command: WILDCARD FIND \<$\D\..00
```

WRITE FILE

format

WRITE FILE *[output-filespec]*

parameter

output-filespec

Optionally, the name of the file to which you want to write the contents of the current buffer. If you do not specify a file, EVE uses the output file specification associated with the buffer. (Typically, this is the same as the input file you specified when you invoked EVE or when you used the GET FILE command.) If there is no file specification associated with the buffer, EVE prompts for a file specification. Pressing RETURN at the prompt cancels the command.

description

Writes the current buffer to a file, without ending your editing session. Specifying a file does not change the buffer name, but may change the output file specification of the buffer for subsequent WRITE FILE commands or for exiting. To find out the file specification of the buffer, use the SHOW command.

example

The following example writes the current buffer to a file named MEMO.TXT:

```
Command: WRITE FILE MEMO.TXT
```

Mail Utility

The VMS Mail Utility (MAIL) allows you to send messages to other users on your system or on any other computer that is connected to your system by means of DECnet–VAX. You can also read, file, forward, delete, print, and reply to messages that other users send to you.

format

MAIL *[filespec] [recipient-name]*

parameters

filespec

Specifies the name of the file to be mailed.

recipient-name

Specifies the name of a user (or users) or a distribution list to which the file is mailed.

When you specify a list of users, separate each name by a comma.

When you specify a distribution list, precede the name of the list with an at sign (@) and enclose both the at sign and the name in quotation marks, as the following example shows:

```
$ MAIL JOKES.DAT "@LIST"
```

usage summary

To use MAIL interactively, enter the following command in response to the DCL prompt:

```
$ MAIL
```

The Mail Utility responds with the following prompt:

```
MAIL>
```

Once MAIL has been invoked, you can enter any of the MAIL commands. To exit from MAIL, enter the EXIT command at the MAIL> prompt.

```
MAIL> EXIT
```

You can also exit from MAIL by pressing CTRL/Z or using the QUIT command.

MAIL Qualifiers

You can supply the /EDIT, /PERSONAL_NAME, /SELF, and /SUBJECT qualifiers when invoking MAIL.

/EDIT

Sets the default to /EDIT for the SEND and REPLY commands and allows you to edit your mail messages.

format

MAIL/EDIT *[=(keyword[=option], . . .)]*

qualifier values

keyword
Allowed keywords are FORWARD, REPLY, and SEND.

option
The EXTRACT option can be used with the REPLY keyword.

example

```
$ MAIL/EDIT
MAIL> SEND
To: EARTH::MAX
Subj: Experiment
   .
   .
   .
[EOB]
*exit
MAIL>
```

This example shows how to use the /EDIT qualifier with the MAIL command enabling you to create and edit a new message. Press CTRL/Z to return to the line-editing prompt (*). Type EXIT to send the message.

/PERSONAL_NAME=name

Specifies the personal name to be used when sending a message. This qualifier does not override the default personal name specified by the SET PERSONAL_NAME command; the personal name is only changed for the current message.

format

MAIL/PERSONAL_NAME *=name file-name recipient-name*
MAIL/NOPERSONAL_NAME

parameter values

name
Personal name to be used. Use quotes around the personal name to include more than one word or to print in lowercase letters.

file-name
Name of file to be sent.

recipient-name
Names of users to whom the message is sent.

example

```
$ MAIL/PERSONAL_NAME ="Joe M." test.dat smith
```

This example shows the user's personal name defined as *Joe M.* in the current message containing the file TEST.DAT sent to user SMITH.

/SELF

Sends a copy of the message containing the file specification on the command line back to you as well as to other users.

format

MAIL/SELF *filespec recipient-name*

parameter values

filespec
Name of file to be sent.

recipient-name
Names of users to whom the message is sent.

example

```
$ MAIL/SELF experiments.dat smith,jones
```

This example shows how to use the /SELF qualifier to send a copy of the message containing the file named EXPERIMENTS.DAT back to you and to users SMITH and JONES.

/SUBJECT

Specifies the subject of the message for the heading. If the text consists of more than one word, enclose the text in quotation marks.

format

MAIL/SUBJECT="text" *filespec recipient-name*

parameter values

filespec
Name of file to be sent.

recipient-name
Names of users to whom the message is sent.

example

```
$ MAIL/SUBJECT="Life in the Big City" newfile.txt JOHNSON
```

This example shows how to use the /SUBJECT qualifier to send a file named NEWFILE.TXT with a subject heading of "Life in the Big City." Use quotation marks around the subject heading to include more than one word or to print in lowercase letters.

MAIL Commands

To enter MAIL commands, first invoke MAIL at the DCL prompt ($) and then enter the MAIL commands at the MAIL> prompt. These commands can be abbreviated to unique, shorter forms (usually as short as one letter). Note that D is the short form of DELETE (not DIRECTORY) and R is the short form of REPLY (not READ).

MAIL provides commands that enable you to do the following:

- Read and organize mail messages.
- Exchange mail messages with other users.
- Remove mail messages.
- Tailor the Mail Utility.
- Exit from MAIL or transfer control to another process while still in MAIL.
- Make hardcopies of mail messages.

BACK

Displays the message preceding the current or last-read message when the last command issued was READ. When the last command issued was DIRECTORY, the BACK command displays the preceding screen of the directory listing.

format

BACK

qualifier

/EDIT

Indicates that the default editor is invoked. You can use the editor to easily peruse the previous message. When you are done, enter the QUIT command. You will see the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (use the EXIT command with the EDT editor) and supply a file name.

COPY

Copies a message to another folder without deleting it from the current folder. If the specified folder does not exist, it is created.

If you want to copy a message to a sequential file (outside of MAIL) instead of to a mail file, use the EXTRACT command.

If you decide (after entering the COPY command, pressing RETURN, and supplying a folder name at the prompt, but before pressing RETURN again) that you do not want to copy the message, press CTRL/C. CTRL/C aborts the operation and keeps you within MAIL.

format

COPY *foldername [filename]*

parameters

foldername

Indicates the name of the folder to which the message is to be copied. If the specified folder does not exist (and you have not entered the qualifier /NOCONFIRM), you are asked whether you want to create it. If you respond with Y, the new folder is created. A folder name can be 1 to 39 characters in length. Valid characters for folder names are A through Z, a through z, dollar sign ($), underscore (_), and 0 through 9.

filename

Indicates the name of the mail file to which the message is to be copied. If the specified mail file does not exist, it is created. If a file name is omitted, the message is copied to the specified folder in the current file.

qualifiers

/ALL

Indicates that all of the currently selected messages are to be copied to another message folder. You select a folder by entering the SELECT command followed by the name of the folder. (See the SELECT command for more information.) If the /ALL qualifier is omitted, only the current message is copied.

/CONFIRM
/NOCONFIRM

Determines whether you will be queried about creating a new folder or file. The default is /CONFIRM.

example

```
MAIL> DIRECTORY
                                                    MAIL
#   From    Date            Subject

1   MARK    29-NOV-1988       Upcoming Meetings
2   GRIM    3-DEC-1988        Horror Stories
3   KATE    7-DEC-1988        Getting a Court for Fridays

MAIL> 2

MAIL> COPY
_Folder: TALES
_File: <RET>
MAIL> SELECT TALES
%MAIL-I-SELECTED, 1 message selected
MAIL> DIRECTORY
                                                    TALES
#   From    Date            Subject

1   GRIM    3-DEC-1988        Horror Stories
```

This example shows how to put a copy of a mail message (from a user named GRIM) into another folder (TALES) and how to move to that folder to see the copy of the mail message.

DELETE

Deletes either the message you are currently reading or the message you just read and moves it to the WASTEBASKET folder. When you enter the EXIT or PURGE commands, your WASTEBASKET folder empties automatically.

To recover a message accidentally deleted (while it is still in the WASTEBASKET folder), select the WASTEBASKET folder, read the desired message, and move it to another folder.

Usually you delete only one mail message at a time, but you may also delete several mail messages using one DELETE command. You may specify a range or a list of messages to be deleted.

format

DELETE *[message-number]*

parameter

message-number
Deletes the message specified by its number, or deletes a range or list of messages.

qualifier

/ALL
Deletes all the currently selected messages. You select a folder by entering the SELECT command followed by the name of the folder. (See the SELECT command for more information.)

example

```
MAIL> DELETE 1,3,5-7,9:11
MAIL>
```

This example shows how to delete mail messages 1, 3, 5, 6, 7, 9, 10, and 11. The hyphen and colon are used to designate a range of numbers.

DIRECTORY

Displays a list of the messages in the current mail file, including message number, sender's name, date, and subject.

You create a new set of selected messages every time you use the following qualifiers:

/BEFORE
/CC_SUBSTRING
/NEW
/SINCE
/MARKED

/FROM_SUBSTRING
/TO_SUBSTRING
/SUBJECT_SUBSTRING

format

DIRECTORY *[foldername]*

parameter

foldername
Specifies the name of the folder. If you omit this parameter and you have already specified a folder, messages from that folder are displayed. If you have not selected a folder, messages from the NEWMAIL folder are displayed. If the NEWMAIL folder does not exist, messages from the MAIL folder are displayed.

qualifiers

/BEFORE=date
Displays a listing of all the mail messages received before the specified date. If no date is specified, a listing of all the mail messages received before the current day ("today") is displayed.

/CC_SUBSTRING=text
Selects messages containing "text" in the CC field of the message.

/EDIT
Invokes the editor using the output of the DIRECTORY command as input to the editor. Enables you to find messages easily by scrolling through the folders or searching text.

/FROM_SUBSTRING=text
Selects messages containing "text" in the FROM field of the message.

/FOLDER
Displays a listing of all the folders contained in the current mail file.

/FULL
Displays the number of records in the message and whether you have replied to the message. External message identification numbers (for messages larger than 3 blocks) are also displayed.

/MARKED
/NOMARKED
Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW
Displays a listing of any new (unread) mail messages. When there are no unread messages, MAIL displays the message "No new messages."

/REPLIED
/NOREPLIED

Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date

Displays a listing of all the mail messages received on or after the specified date. If no date is specified, a listing of all the mail messages received on the current day ("today") is displayed.

/START=start-point

Indicates the first message number you want to display. For example, to display all the messages beginning with number three, enter the command line DIRECTORY/START=3. Use the /START qualifier with the /FOLDER qualifier to indicate the first folder name you want to display. For example, to display all the folder names alphabetically following PLEAT, enter the command line DIRECTORY/START=PLEAT/FOLDER.

/SUBJECT_SUBSTRING=text

Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text

Selects messages containing "text" in the TO field of the message.

example

```
MAIL> DIRECTORY/SUBJECT_SUBSTRING= POUND
                                                    MAIL
#   From    Date            Subject

1   BILL    13-APR-1988       The Pound
```

This example shows how to use the /SUBJECT_SUBSTRING qualifier with the DIRECTORY command to find messages that contain the substring POUND.

```
MAIL> DIRECTORY/FOLDER
Listing of folders in DISK$:[BACON]MAIL.MAI;1
    Press CTRL/C to cancel listing
MAIL                                NEW_HIRES
PROJECTS                            SALES_LEADS
```

This example shows how to display a listing of all the folders in the current mail file.

EXIT

Allows you to exit from MAIL. You can also exit from MAIL by pressing CTRL/Z.

format

EXIT

EXTRACT

Places a copy of the current message into a sequential file. If you want to copy a mail message to a folder in an indexed sequential mail file, use either the COPY, FILE, or MOVE command.

format

EXTRACT *filespec*

parameter

filespec
Specifies the name of the output file to which the message is copied. The default file type is TXT. By default, the device and directory matches your current default device and directory.

qualifiers

/ALL
Copies all the currently selected messages to the specified file. Each message is separated by a form feed.

/APPEND
Adds the selected message to the end of the specified file. If the file does not exist, it is created. When you do not specify /APPEND, MAIL creates a new sequential file.

/MAIL
Specifies that the output file be a sequential mail file with a default file type of MAI and a protection code of (S:RW,O:RW,G,W). By default, the protection codes of the device and directory match those of your mail file directory. Like /APPEND, /MAIL adds the selected message to the end of the specified file.

/NOHEADER
Removes the header information (To: CC: From: Subject:) from the mail message.

example

```
MAIL> EXTRACT/ALL/NOHEADER
_File: OUTER.DAT
%MAIL-I-CREATED, DISK$MEGAWORK:[CROWN]OUTER.DAT;1 created
MAIL>
```

This example shows how to place a copy of all the messages in the currently selected folder into a sequential file called OUTER.DAT. The /NOHEADER qualifier prevents the header information from being copied.

FILE

Moves the current message to the specified folder. You can use the FILE command and the MOVE command interchangeably because they work the same way. (Note, however, that the FILE command deletes the message from the original folder, unlike the COPY command, which leaves a copy.)

If (after entering the FILE command, pressing RETURN, and supplying a folder name at the prompt, but before pressing RETURN again) you decide that you do not want to file the message, press CTRL/C. CTRL/C aborts the operation and keeps you within MAIL.

format

FILE *foldername [filename]*

parameters

foldername

Indicates the name of the folder to which the current message is to be moved. If the specified folder does not exist, you are asked whether you want to create it. If you respond with Y, the new folder is created.

A folder name can be 1 to 39 characters in length. Valid characters for folder names are A through Z, a through z, dollar sign ($), underscore (_), and 0 through 9.

filename

Indicates the name of the mail file to which the current message is to be moved. If the file name is omitted, the message is moved to the specified folder in the current file.

qualifiers

/ALL

Moves all the messages in the current folder to the specified folder.

/CONFIRM
/NOCONFIRM

Determines whether you are queried about creating a new folder or file. The default is /CONFIRM.

example

```
MAIL> 2
MAIL> FILE ❶
_Folder: WINNERS ❷
_FILE: <RET> ❸
Folder WINNERS does not exist.
Do you want to create it (Y/N, default is N)? y
%MAIL-I-NEWFOLDER, folder WINNERS created
MAIL> SELECT WINNERS ❹
%MAIL-I-SELECTED, 1 message selected
MAIL> DIRECTORY ❺
                                                WINNERS

#   From    Date            Subject

1   BURK    18-APR-1988       Early American Art

MAIL>
```

❶ Enter the FILE command to move the current message to a new folder.

❷ Specify a name for the new folder.

❸ Press RETURN to retain the default file.

❹ To move to the new folder, enter the SELECT command followed by the name of the new folder (WINNERS).

❺ Enter the DIRECTORY command to see the transferred message in the newly created folder (WINNERS).

This example shows how to FILE a message in a new folder named WINNERS.

FORWARD

Sends a copy of the message you are currently reading (or have just read) to one or more users. MAIL prompts you for the name of the user or users to whom you want to forward the message.

If you change your mind about forwarding a message after you have already entered the FORWARD command, press CTRL/C to abort the message. The MAIL> prompt is displayed.

format

FORWARD

qualifiers

/CC_PROMPT
/NOCC_PROMPT
Prompts for CC: line in the mail header. Overrides the SET CC_PROMPT command.

/EDIT
Determines whether the default editor is invoked to edit the message you are forwarding.

/NOHEADER
Enables you to forward a message without the original header information supplied from the user that sent it. The default is /HEADER.

/PERSONAL_NAME=name
/NOPERSONAL_NAME
Specifies the personal name to be used when forwarding the message. Overrides the default personal name specified with the SET PERSONAL_NAME command for this message only. The /NOPERSONAL_NAME qualifier sends a message with a null personal name field.

/SELF
/NOSELF
Specifies that a copy of the forwarded message is to be sent to you. Overrides the SET COPY_SELF command.

/SUBJECT="subject-text"
Prompts for the subject of the mail message to be sent.

example

```
MAIL> 3
From:       PRESTON
To:         MARLEY
Subj:       Snakes

Beasts, under the earth, crawling...

MAIL> FORWARD/NOHEADER
To: SOUND::BURTON
Subj: Snakes Again
   .
   .
   .
MAIL> READ
From:       MARLEY
To:         SOUND::BURTON
Subj:       Snakes Again
```

```
Beasts, under the earth, crawling...
```

This example shows how to forward a message to a user (SOUND::BURTON) without the original header information (From: PRESTON, To: MARLEY, Subj: Snakes).

HELP

Enables you to obtain information about the Mail Utility.

To obtain information about all of the MAIL commands, enter the following command:

```
MAIL> HELP
```

To obtain information about individual commands or topics, enter HELP followed by the command or topic name.

format

HELP *[topic]*

parameter

topic
Indicates a topic about which you want information. To display the list of available topics, enter the HELP command at the MAIL> prompt.

MARK

The MARK command sets a flag in the message header setting the current or message-identification message as marked. Marked messages are displayed with an asterisk (*) in the left-hand column of the directory listing. A marked message can serve as a reminder of important information. The /ALL qualifier sets all currently selected messages as unmarked.

The UNMARK command clears a flag in the message header setting the current or message-id message as unmarked (asterisk is deleted).

format

MARK *[/ALL] [message-number]*
UNMARK *[/ALL] [message-number]*

parameter

message-number
Indicates the message number to be marked or unmarked.

qualifier

/ALL

Sets all currently selected messages as marked.

example

```
MAIL> DIR MISC
  #          From                Date            Subject

  1        MARS::SMITH          13-AUG-1988     Training Information
  2        JUPITER::COLLINS      22-AUG-1988     Ideas
  3        JUPITER::PETERS       24-AUG-1988     Meeting
MAIL> MARK 2,3
MAIL> DIR
  #          From                Date            Subject

  1        MARS::SMITH          13-AUG-1988     Training Information
* 2        JUPITER::COLLINS      22-AUG-1988     Ideas
* 3        JUPITER::PETERS       24-AUG-1988     Meeting
```

In this example, messages 2 and 3 in folder MISC are marked with an asterisk.

MOVE

The MOVE command is synonymous with the FILE command.

NEXT

Skips to the next message and displays it. This command is useful if, while reading through your messages, you encounter a long message that you would like to skip over.

format

NEXT

qualifier

/EDIT

Invokes your default editor. You can use your editor to peruse the next message. When you are done, enter the QUIT command. You see the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (enter the EXIT command with the EDT editor) and supply a file name.

PRINT

Adds a copy of the message you are currently reading to the print queue. The files created by the PRINT command are not actually released to the print queue until you exit from MAIL, so that multiple messages are concatenated into one print job (unless the /NOW or /PRINT qualifier is specified).

format

PRINT

qualifiers

/AFTER=time
Requests that the job not be printed until a specific time of day. You can specify either absolute or delta time.

/ALL
Indicates that all the currently selected messages be printed.

/BURST=keyword
/NOBURST
Controls whether a burst page is printed preceding a message. The /BURST qualifier can take either of two keywords: ALL or ONE. The ALL keyword indicates that each file in the job is to be preceded by a burst page and flag page. The ONE keyword indicates that a burst page applies only to the first copy of the first file in the job.

/CANCEL
Cancels all messages that have been queued for printing during this session.

/COPIES=n
Indicates the number of copies of the print job to be printed.

/FEED
/NOFEED
Controls whether the PRINT command automatically inserts form feeds when it nears the end of a page. /FEED is the default.

/FLAG=keyword
/NOFLAG
Controls whether a flag page is printed preceding a message. The /FLAG qualifier can take either of two keywords: ALL or ONE. The ALL keyword indicates that each file in the job is preceded by a flag page. The ONE keyword indicates that a flag page applies only to the first copy of the first file in the job.

/FORM=form-name
Specifies the name or number of the form that you want for the print job. Enter the DCL command SHOW QUEUE/FORM to list the available forms.

/HOLD
/NOHOLD
Controls whether the message is available for print immediately. The print job is not released for actual printing until you use the DCL command SET QUEUE/ENTRY/RELEASE to release it.

/NAME=job-name
Defines the name string to identify the job.

/NOTIFY
Indicates that you are to be notified by a broadcast message when the file or files have been printed. /NONOTIFY is the default.

/NOW
Sends all messages that have been queued for printing with the PRINT command during this session to the printer. Allows the job to print without exiting mail. This qualifier is synonymous with the /PRINT qualifier.

/PARAMETERS=(parameter[, . . .])
Specifies from one to eight optional parameters to be passed to the job.

/PRINT
The /PRINT qualifier is synonymous with the /NOW qualifier.

/QUEUE=queue-name
The name of the queue to which the message is to be sent. If the /QUEUE qualifier is not specified, the message is queued to the SYS$PRINT printer. If you enter the PRINT command more than once and specify a different queue name, any previously queued messages are released to that print queue.

/SPACE
/NOSPACE
Controls whether output is double-spaced.

/TRAILER=keyword
/NOTRAILER
Controls whether a trailer page is printed at the end of the message. The /TRAILER qualifier can take either of two keywords: ALL or ONE. The ALL keyword indicates that each file in the job is preceded by a trailer page. The ONE keyword indicates that a trailer page applies only to the last copy of the last file in the job.

example

```
MAIL> 5
MAIL> PRINT/QUEUE=LMNO
MAIL> EXIT
  Job MAIL (queue LMNO_PRINT, entry 333) started on LMNO_PRINT

$
```

This example shows how to add message number 5 to queue LMNO_PRINT.

READ

Displays your messages.

The READ command can be entered with or without parameters. Pressing the RETURN key is the same as entering the READ command without parameters. If you press RETURN immediatedly after MAIL is invoked, MAIL displays the first unread message in your NEWMAIL folder. If all messages have been read or you have no new messages, MAIL displays the first message in the MAIL folder. Each time you enter the READ command without parameters or press RETURN, MAIL displays the next message.

If a new message arrives while you are in MAIL, you can enter READ/NEW to read the message, and then return to the previous MAIL activity.

You create a new set of selected messages every time you use the following qualifiers:

/BEFORE
/CC_SUBSTRING
/NEW
/SINCE
/MARKED
/FROM_SUBSTRING
/TO_SUBSTRING
/SUBJECT_SUBSTRING

format

READ *[foldername] [message-number]*

parameters

foldername
Specifies the name of the folder. If you omit this parameter and you have already specified a folder, messages from that folder are displayed. If you have not selected a folder, messages from the NEWMAIL folder are displayed. If the NEWMAIL folder does not exist, messages from the MAIL folder are displayed.

message-number
Indicates the number of the message to be read. The message number represents the position of a message in a folder. If you specify a number greater than the number of messages in the folder, MAIL displays the last message in the folder. Therefore, to read the latest message in a folder, specify a large message number or enter the LAST command.

qualifiers

/BEFORE=date

Displays mail messages received before the specified date. If no date is specified, all the mail messages received before the current day ("today") are displayed.

/CC_SUBSTRING=text

Selects messages containing "text" in the CC field of the message.

/EDIT

Invokes the default editor. You can use the editor to easily peruse the next message. When you are done, enter the QUIT command and return to the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (use the EXIT command with the EDT editor) and supply a file name.

/FROM_SUBSTRING=text

Selects messages containing "text" in the FROM field of the message.

/MARKED
/NOMARKED

Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW

Displays new mail messages received while you are in MAIL. If there are no new messages, the message "No new messages" is displayed.

/REPLIED
/NOREPLIED

Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date

Displays mail messages received on or after the specified date. If no date is specified, all the mail messages received after the current day ("today") are displayed.

/SUBJECT_SUBSTRING=text

Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text

Selects messages containing "text" in the TO field of the message.

example

```
MAIL> READ/SUBJECT_SUBSTRING= MEETINGS
MAIL
#   From    Date          Subject

1   BILL    16-APR-1988     Future Meetings
```

This example shows how to use the /SUBJECT_SUBSTRING qualifier with the READ command to find messages that contain the substring MEETINGS.

REPLY

The REPLY command is synonymous with the ANSWER command.

SEARCH

Searches the currently selected folder for the message containing the first occurrence of the specified text string.

format

SEARCH *search-string*

parameter

[search-string]
Indicates the text string that MAIL searches for in the currently selected messages. The search starts from the beginning of the messages in the current folder. If **search-string** is not specified, a search is made for the previously specified string, starting after the message you are currently reading (or have just read).

example

```
MAIL> SEARCH "under the"

From:    BURT
To:      ANTON
Subj:    Coal Mines

They commute under the earth...

MAIL>
```

This example shows how to search for the string *under the*.

SELECT

Establishes a set of messages that you can manipulate. You can copy or move selected messages from one folder to another; or you can read and delete, or search and extract, a set of messages. After you select a set of messages, you can use the following commands to affect them:

COPY
DELETE
DIRECTORY
EXTRACT
FILE
MOVE
READ
SEARCH

You can also use the SELECT command to move from one folder to another. If you select a folder that does not exist, MAIL displays the following message:

```
%MAIL-E-NOTEXIST, folder "foldername" does not exist
```

format

SELECT *[foldername]*

parameter

foldername

Specifies the name of the folder. If you omit this parameter and have already specified a folder, messages from that folder are selected. If you have not specified a folder, messages from the NEWMAIL folder are selected. If the NEWMAIL folder does not exist, messages from the MAIL folder are selected.

qualifiers

/BEFORE=date

Indicates that messages dated before the specified date be selected. If no date is specified, all the messages received before the current day ("today") are selected.

/CC_SUBSTRING=text

Selects messages containing "text" in the CC field of the message.

/FROM_SUBSTRING=text

Selects messages containing "text" in the FROM field of the message.

/MARKED
/NOMARKED

Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW
Indicates that new (unread) messages be selected. When a mail file other than your default mail file is open, MAIL closes the file and opens your default mail file.

/REPLIED
/NOREPLIED
Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date
Indicates that messages dated after the specified date be selected. If no date is specified, all the messages received on the current day ("today") are selected.

/SUBJECT_SUBSTRING=text
Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text
Selects messages containing "text" in the TO field of the message.

example

```
MAIL> DIRECTORY/FOLDERS ❶
Listing of folders in DISK$APEX:[HARRIS]MAIL.MAI;1
Press CTRL/C to cancel listing

MAIL                NEWMAIL
WASTEBASKET         JUNK
COURSES

MAIL> SELECT WASTEBASKET ❷
%MAIL-I-SELECTED, 3 messages selected
MAIL> DIRECTORY ❸
                                    WASTEBASKET
#   From    Date                Subject

1   MORRIS  19-APR-1988           Venus Fly Traps
2   MORRIS  21-APR-1988           The Aloe
3   BURT    22-APR-1988           Scales
```

❶ Enter the DIRECTORY/FOLDERS command to display all currently existing folders.

❷ Enter the SELECT command to move to the WASTEBASKET folder.

❸ Enter the DIRECTORY command to display the contents of the WASTEBASKET folder.

This example shows how to use the SELECT command to move from the MAIL folder to the WASTEBASKET folder.

SEND

Sends a message to one or more other users. You can use the SEND command and the MAIL command interchangeably because they work the same way.

MAIL prompts you first for the name of the user or users to receive the message. You reply with the user names or with the file names of distribution lists, in the following format:

[[nodename::]username,...] [,] [@listname[,...]]

If you have entered the SET CC_PROMPT command, you can specify names of users to receive carbon copies of the message at the CC: prompt.

Next, MAIL prompts you for the subject of the mail. To avoid the Subj: prompt, specify the /SUBJECT qualifier with the SEND command.

You can include a file specification with the SEND command. If you do, the text in that file is sent to the specified users. If you do not specify a file, MAIL prompts you for the text of your message.

Enter the message you want to send, then press CTRL/Z. Note that, once you have typed a line and pressed RETURN, there is no way to edit it. Using the /EDIT qualifier enables you to edit the entire message before you send it. The /LAST qualifier enables you to send the last message. The /LAST qualifier, used with the /EDIT qualifier, enables you to edit the last message you sent. If you decide not to send a message you are typing but want to stay within the Mail Utility, press CTRL/C to abort the message. You then receive the MAIL> prompt. CTRL/Y exits you from MAIL.

format

SEND *[filespec]*

parameter

filespec
Indicates the name of the file to be sent.

qualifiers

/CC_PROMPT
/NOCC_PROMPT
Prompts for CC: line in the mail header. Overrides the SET CC_PROMPT command.

/EDIT
/NOEDIT
Determines whether the default editor is invoked to edit the message you are sending. The /NOEDIT qualifier overrides the SEND/EDIT default if you entered the DCL command MAIL/EDIT.

If you are interrupted while editing a mail message, a journal file is created containing your edits. To recover your edits, enter the following command line. (You may substitute another word of your choice for the word OOPS.)

```
$ EDIT/RECOVER/JOURNAL=SYS$SCRATCH:MAIL.JOU SYS$SCRATCH:OOPS.TMP
```

The editor is invoked displaying the text of the message you were editing. After you exit from the editor, you can mail the file (in this case, OOPS.TMP) by using the MAIL command SEND, as follows:

```
MAIL> SEND OOPS.TMP
To: MCNALLY
Subject: Vacationing in Venice
```

/LAST

Specifies that the last message be used as the text for the message you are currently sending. You can use /LAST with the /EDIT qualifier to edit the message before sending it. Press CTRL/C and enter the following command:

```
MAIL> SEND/LAST/EDIT
```

This command invokes the editor and allows you to edit the last message. Send the revised message by entering the EXIT command.

/PERSONAL_NAME=name
/NOPERSONAL_NAME

Specifies the personal name to be used when sending this message. The /NOPERSONAL_NAME qualifier sends a message with a null personal name field.

/SELF
/NOSELF

Determines whether MAIL sends a copy of the message you are sending back to you. The /NOSELF qualifier overrides the SET COPY_SELF SEND command.

/SUBJECT="subject-text"

Specifies the subject of the mail message to be sent.

example

```
MAIL> SEND
To: FLIGHT::WRIGHT
Subj: Meeting
Enter your message below.  Press CTRL/Z when complete, CTRL/C to quit:
We will have our meeting on Monday, August 31st, as scheduled.
Please make sure you are prompt.
    .
    .
    .
CTRL/C
% MAIL_E_SENDABORT, no message sent
MAIL> SEND/LAST/EDIT
To: FLIGHT::WRIGHT
Subj: Meeting date correction
We will have our meeting on Friday, September 4th, as scheduled.
Please make sure you are prompt.
    .
    .
    .
[EOB]
CTRL/Z
 EXIT
MAIL>
```

This example shows how to edit the last message before sending it to user WRIGHT on node FLIGHT. To make a change in text, enter CTRL/C and invoke the editor by entering the SEND/LAST/EDIT command. Edit the message you were in the process of entering, and send it by entering the EXIT command.

SET/SHOW CC_PROMPT

Sets the default for determining whether the carbon copy (CC:) prompt appears when sending a message.

format

SET CC_PROMPT
SET NOCC_PROMPT
SHOW CC_PROMPT

example

```
MAIL> SET CC_PROMPT
MAIL> SEND
To:Smith
CC:Jones
Subject:
```

This example shows how to set the carbon copy prompt. A copy of the message is sent to JONES.

SET/SHOW EDITOR

Invokes a text editor. Use the MAIL command SEND/EDIT to edit the message. The SHOW EDITOR command displays the name of the editor.

format

SET EDITOR *editor-name*
SHOW EDITOR

parameter

editor-name
Indicates the name of the editor. You can use any callable editor available on your system.

qualifiers

None.

example

```
MAIL> SHOW EDITOR
Your editor is EDT

MAIL> SET EDITOR TPU
MAIL> SHOW EDITOR
Your editor is TPU

MAIL> SEND/EDIT
To: WHITE::STAFFORD
Subject: Manufacturing Office
```

This example shows how to change the editor from the default EDT editor to the TPU editor. Enter the MAIL command SEND/EDIT to edit the text of a message. Send the message by pressing CTRL/Z.

SET/SHOW COPY_SELF

Sets the default for determining whether the SEND, REPLY, or FORWARD commands return to the sender a copy of the message being sent.

By specifying NOSEND, NOREPLY, or NOFORWARD with the SET COPY_SELF command, you can clear any default copying you have established with the SET COPY_SELF command.

The SHOW COPY_SELF command displays which command (SEND, REPLY, or FORWARD) automatically sends a copy of the message to you.

format

SET COPY_SELF *command [,command]*
SHOW COPY_SELF

parameter

command
The **command** parameter can be any one of the following: SEND, NOSEND, REPLY, NOREPLY, FORWARD, or NOFORWARD. You can use NOSEND, NOREPLY, and NOFORWARD to reverse previous settings of SEND, REPLY, or FORWARD.

example

```
MAIL> SET COPY_SELF SEND
MAIL> SHOW COPY_SELF
Automatic copy to yourself on SEND
```

This example shows how to use the SET COPY_SELF command to enable copies of mail messages you send to be returned back to you. The SHOW COPY_SELF command indicates that you have enabled automatic copying when you enter a SEND command.

SET/SHOW FORWARD

Sets a forwarding address for your mail. After you enter the SET FORWARD command, the address you specify will receive mail messages.

The default you establish with the SET FORWARD command remains in effect until you enter the SET NOFORWARD command.

The SHOW FORWARD command displays the name of the specified forwarding address.

If you have SYSNAM privilege, you can set and show forwarding addresses for other users.

format

SET FORWARD *address*
SET NOFORWARD
SHOW FORWARD

parameter

address
Indicates the address (NODE::NAME) to which your mail is forwarded.

qualifiers

/ALL
The /ALL qualifier lists forwarding information or displays a message if the specified user does not have forwarding enabled.

/USER=user-name
Indicates the name of another user for whom you are setting or showing a forwarding address. You can use the /USER qualifier only if you have SYSNAM privilege. With the SHOW FORWARD command, there are two ways to show a user's forwarding address: You can specify the user name, or you can use the wildcard characters (* or %) to search for names with a particular string in common.

example

```
MAIL> SET FORWARD NEXUS::LARS
MAIL> SHOW FORWARD
Your mail is being forwarded to NEXUS::LARS

MAIL>
```

This example shows how a user named LARS establishes a forwarding address on node NEXUS with the SET FORWARD command, and displays the forwarding address with the SHOW FORWARD command.

SET/SHOW PERSONAL_NAME

Enables you to append a field to the end of the From: field of mail messages you send. You can fill this field with your full name or any other information.

The SET NOPERSONAL_NAME command clears any name you previously specified with the SET PERSONAL_NAME command.

The SHOW PERSONAL_NAME command displays a user's personal name.

format

SET PERSONAL_NAME *"text-string"*
SET NOPERSONAL_NAME
SHOW PERSONAL_NAME

parameter

"text-string"
Specifies the string for the From: field of mail messages you send. You must enclose the string in quotation marks; otherwise, MAIL converts it to uppercase letters. You must begin the string with an alphabetic character and avoid two consecutive embedded spaces within the string. The length of the text string should not exceed 127 characters.

qualifiers

/ALL

The /ALL qualifier lists personal name information or displays a message if the specified user has not entered a personal name.

/USER=name

Used with the SHOW PERSONAL_NAME command to allow a user with SYSNAM privilege to list personal names set by other users. There are two ways to show a user's personal name. The user name can be specified, or you can use the wildcard characters (* or %) to search for names with a particular string in common.

example

```
MAIL> SET PERSONAL_NAME "Catherine the Great"
    .
    .
    .
MAIL> SEND
    .
    .
    .
New mail on node FLAXEN from ALPHA::BELLINI "Catherine the Great"
    .
    .
    .
From: ALPHA::BELLINI "Catherine the Great"  19-APR-1988 15:34
To: FLAXEN::STARCK
```

This example shows how a user named BELLINI sets her personal name to Catherine the Great.

Sort/Merge Utility

The VMS Sort/Merge Utility sorts records or merges input files. To sort one or more input files, specify the SORT command. These files are sorted according to the fields you select and one reordered output file is generated. To merge up to 10 input files that have previously been sorted according to the same key fields, specify the MERGE command. One output file is generated.

format

SORT *input-files output-file*

parameters

input-files

Specifies the files to be sorted. Up to 10 input files can be sorted to create one output file. Multiple input file specifications must be separated by commas. The default file type is DAT.

output-file

Specifies the file to be created. Only one file can be specified. If you omit a file type in the file specification, SORT defaults to the file type of the first input file.

format

MERGE *input-files output-file*

parameters

input-files

Specifies the files to be merged. Up to ten files, presorted by the same keys, can be specified. Multiple file specifications must be separated by commas. The default file type is DAT.

output-file

Specifies the merged file to be created. Only one output file can be specified. If you omit a file type in the file specification, MERGE defaults to the file type of the first input file.

usage summary

Both the SORT and MERGE commands invoke the VMS Sort/Merge Utility. After completing an operation, the Sort/Merge Utility exits and returns the user to DCL command level. You can specify where you want the results of a SORT/MERGE operation with the output file parameter.

SORT/MERGE Qualifiers

This section describes the qualifiers to the SORT and MERGE commands. These qualifiers enable you to define your key fields, to describe the data in those fields, and to specify various SORT options.

/CHECK_SEQUENCE

Verifies the sequence of the records in MERGE input files. By default, the sequence of records is checked.

format

/CHECK_SEQUENCE
/NOCHECK_SEQUENCE

example

```
$ MERGE/KEY=(SIZE:4,POSITION:3)/NOCHECK_SEQUENCE  PRICE1.DAT -
_$ PRICE2.DAT PRICE.LIS
```

The /NOCHECK_SEQUENCE qualifier specifies that sequence of the input files, PRICE1.DAT and PRICE2.DAT, need not be checked because the records in those files were sorted on the same key and the sequence of records is correct.

/COLLATING_SEQUENCE

Selects one of three predefined collating orders for character key fields, or specifies the name of a National Character Set (NCS) collating sequence to be used in comparing character keys. SORT arranges characters in ASCII sequence by default; the EBCDIC and MULTINATIONAL sequences can also be used.

format

/COLLATING_SEQUENCE=type
/COLLATING_SEQUENCE=cs-name

parameters

ASCII
Arranges characters according to ASCII sequence. ASCII is the default sequence and need not be specified.

EBCDIC
Arranges characters according to EBCDIC sequence. The characters remain in ASCII representation; only the order is changed.

MULTINATIONAL

Arranges characters according to MULTINATIONAL sequence, which collates the international character set. When you use the MULTINATIONAL sequence, characters are ordered according to the following rules:

- All diacritical forms of a character are given the collating value of the character (A′, A″, A` collate as A).
- Lowercase characters are given the collating value of their uppercase equivalents (a collates as A, a″ collates as A″).
- If two strings compare as equal, tie-breaking is performed. The strings are compared to detect differences due to diacritical marks, ignored characters, or characters that collate as equal although they are actually different. If the strings still compare as equal, another comparison is done based on the numeric codes of the characters. In this final comparison, lowercase characters are ordered before uppercase.

Care should be taken when sorting or merging files for further processing using the MULTINATIONAL sequence. Sequence checking procedures in most programming languages compare numeric characters. Because MULTINATIONAL is based on actual graphic characters and not on the codes representing those characters, normal sequence checking does not work.

Note that, some languages do not support MULTINATIONAL comparisons and instead can use the LIB$COMPARE_MULTI routine.

CS-NAME

Arranges character keys according to the named sequence, which must be a collating sequence defined in a VAX NCS library.

example

```
$ SORT/COLLATING_SEQUENCE=MULTINATIONAL -
_$ NAMES.DAT,NOM.DAT LIST.LIS
```

This SORT command arranges the input files NAMES.DAT and NOM.DAT according to the MULTINATIONAL collating sequence to create the output file LIST.LIS.

/DUPLICATES

Eliminates all but one of the multiple records with duplicate keys. By default, SORT retains multiple records with duplicate keys.

format

/DUPLICATES
/NODUPLICATES

example

```
$ SORT/KEY=(POSITION:3,SIZE:5,DECIMAL)/NODUPLICATES -
_$ ACCT1,ACCT2 ACCT.LIS
```

This SORT command arranges the two input files according to the key supplied and eliminates all but one of the multiple records with equal keys.

/KEY

Describes key fields, including the position, size, sorting order, and data type. By default, SORT reorders a file by sorting entire records with character data in ascending order. Any other type of key field must be specified. When specifying multiple keys, use a separate /KEY qualifier for each key.

format

/KEY=(field [,...])

qualifier values

POSITION:n
Specifies the position of the first byte in the key field. A value of 1 to 32,767 may be specified. The first byte in a record is considered position 1. Both the position and the size of the key field must be specified. If a decimal sign is stored in a separate byte in the key field, that byte is counted when you determine position.

SIZE:n
Specifies the length of the key field. Both the position and size of the key field must be specified. The total composite size of all keys and the original input record length must be less than 32,767 bytes. If the decimal sign is stored in a separate byte in the key field, that byte is not counted toward the size of the data.

The data type of the key determines what values are acceptable when specifying size:

- 1 to 32,767 characters for character data
- 1, 2, 4, 8, or 16 bytes for binary data

- 1 to 31 digits for decimal data
- No value is necessary for floating point data

ASCENDING	Orders the sorting operation in ascending alphabetical or numerical order. ASCENDING is the default order.
DESCENDING	Orders the sorting operation in descending alphabetical or numerical order.
CHARACTER	Specifies character data in the key field. CHARACTER is the default data type.
BINARY	Specifies binary data in the key field.
SIGNED	Specifies signed binary or decimal data in key field. SIGNED is the default for binary and decimal data.
UNSIGNED	Specifies unsigned binary or decimal data in the key field.
F_FLOATING	Specifies F_FLOATING format data in the key field.
D_FLOATING	Specifies D_FLOATING format data in the key field.
G_FLOATING	Specifies G_FLOATING format data in the key field.
H_FLOATING	Specifies H_FLOATING format data in the key field.
DECIMAL	Specifies decimal data in the key field.
TRAILING_SIGN	Specifies trailing sign decimal data in the key field. TRAILING_SIGN is the default for decimal data.
LEADING_SIGN	Specifies leading sign decimal data in the key field.
OVERPUNCHED_SIGN	Specifies overpunched decimal data in the key field. OVERPUNCHED_SIGN is the default for decimal data.
SEPARATE_SIGN	Specifies separate sign decimal data in the key field.
ZONED	Specifies zoned decimal data in the key field.
PACKED_DECIMAL	Specifies packed decimal data in the key field.
NUMBER:n	Specifies the order of priority of each key if you do not list multiple keys in the order of their priority. A value of 1 to 255 may be specified.

example

```
$ SORT/KEY=(POS:16,SIZ:3)/KEY=(POS:1,SIZ:11) -
_$ /KEY=(POS:40,SIZ:2,DESC) YRENDAVG.DAT YRAVGSRT.LIS
```

This SORT command identifies three key fields. The input file, YRENDAVG, is first sorted by the key beginning in position 16, then by the key beginning in position 1, and finally by the key beginning in position 40. The third key used sorts in descending order.

/PROCESS

Defines the internal sorting process. The /PROCESS qualifier allows you to choose one of four processes: record, tag, address, or index. By default, SORT uses a record sorting process. Use only with the SORT command.

format

/PROCESS=type

qualifier values

RECORD
Keeps records intact while sorting and produces an output file consisting of complete records. Record is the default sorting process.

TAG
Sorts only the keys and then rereads the input file to produce an output file consisting of complete records.

ADDRESS
Sorts only the keys and produces an output file that is an index of record addresses in binary format. The index must be submitted to a program for further processing.

INDEX
Sorts only the keys and produces an output file that is an index of keys and of record addresses in binary form. The index must be submitted to a program for further processing.

example

```
$ SORT/KEY=(POS:40,SIZ:2,DESC)/PROCESS=TAG YRENDAVG.DAT-
_$ DESCYRAVG.LIS
```

This SORT operation uses a tag sorting process to create the output file, DESCYRAVG.LIS.

/SPECIFICATION

Identifies a SORT or MERGE specification file.

format

/SPECIFICATION=filespec

qualifier value

filespec

Specifies the SORT/MERGE specification file. The default file type is SRT.

example

```
$ SORT/SPECIFICATION=ACCTS.SRT SALES1.DAT,SALES2.DAT MAILING.LIS
```

This SORT command arranges the input files according to the instructions detailed in the specification file, ACCTS.SRT.

/STABLE

Directs records with equal keys to the output file in their input file order. The default condition is /NOSTABLE.

format

/STABLE

/NOSTABLE

example

```
$ SORT/KEY=(POS:1,SI:5,DECIMAL)/STABLE PRICESA.DAT,PRICESB.DAT -
_$ PRICESC.DAT SUMMARY.LIS
```

In this SORT operation, records with equal keys from PRICESA.DAT will be listed first, followed by those from PRICESB.DAT, followed by those from PRICESC.DAT.

/STATISTICS

Displays a statistical summary that can be used for optimization.

format

/STATISTICS

example

```
$ SORT /STATISTICS PRICE1.DAT,PRICE2.DAT PRICE.LIS
```

This SORT command results in the following statistical display:

```
VMS Sort/Merge Statistics

Records read:          793        Input record length:        80
Records sorted:        793        Internal length:            80
Records output:        793        Output record length:       80
Working set extent:    100        Sort tree size:            412
Virtual memory:        433        Number of initial runs:      2
Direct I/O:             22        Maximum merge order:         2
Buffered I/O:            9        Number of merge passes:      1
Page faults:          3418        Work file allocation:      114
Elapsed time: 00:00:05.98         Elapsed CPU:       00:00:03.63
```

In the sample statistics display, the sort data structure size is limited by the small working set extent. By doubling the working set extent, you can almost double the sort data structure size, enabling all the records to fit in memory without using work files.

/WORK_FILES

Used for optimization.

format

/WORK_FILES=n

qualifier value

n

Specifies the number of work files requested; 0 to 10 files may be specified.

example

```
$ ASSIGN DRA5: SORTWORK0
$ ASSIGN DB0: SORTWORK1
$ ASSIGN DB1: SORTWORK2
$ SORT/KEY=(POS:1,SI:80)/WORK_FILES=3 -
_$ STATS1,STATS2,STATS3,STATS4 SUMMARY.LIS
```

Since the input files in this sort operation are large files, specifying three work files improves the efficiency of the sort operation.

Input File Qualifiers

This section describes the qualifier to the input file. This qualifier should be specified immediately after the input file specification.

/FORMAT

Input File Qualifier

Defines input file characteristics; allows you to specify or override record or file size.

format

input filespec/**FORMAT=***(type:n,[...])*

qualifier values

RECORD_SIZE:n

Specifies the file's longest record length (LRL) in bytes.

The maximum longest record length that can be specified depends on the file organization:

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes for variable records with fixed-length control (VFC) format.

FILE_SIZE:n

Specifies input file size in blocks. The maximum file size accepted is 4,294,967,295 blocks.

Output File Qualifiers

This section describes the qualifiers to the output file. These qualifiers should be specified immediately after the output file specification.

/ALLOCATION

Output File Qualifier

Used for optimization.

format

output filespec/**ALLOCATION=n**

qualifier value

n
Specifies the number of blocks to be allocated. A value of 1 to 4,294,967,295 is allowed.

/BUCKET_SIZE

Output File Qualifier

Used for optimization.

format

output filespec/**BUCKET_SIZE=n**

qualifier value

n
Specifies the bucket size. A value of 1 to 32 is allowed.

/CONTIGUOUS

Output File Qualifier

Used for optimization.

format

output filespec/**CONTIGUOUS**

/FORMAT

Output File Qualifier

Specifies the output file record format if it differs from the input file format.

format

output filespec/**FORMAT=***(type:n ...)*

qualifier values

FIXED:n
Specifies fixed-length records in the output file.

VARIABLE:n
Specifies variable-length records in the output file.

CONTROLLED:n
Specifies variable with fixed-length control (VFC) records in the output file.

n
Optionally indicates the maximum record size (in bytes) of the output records. The maximum record size allowed depends on the file organization.

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes. If you do not specify the maximum record size, the default is a length large enough to hold the longest output record.

SIZE:n
Specifies the size, in bytes, of the fixed portion of VFC (CONTROLLED) records, up to a maximum of 255 bytes. If you do not specify SIZE, the default is the size of the fixed portion of the first input file. If you specify this size as 0, RMS defaults the value to 2 bytes.

BLOCK_SIZE:n
Specifies the output file's block size, in bytes, if you have directed the file to magnetic tape. You can also accept the default. If the input file is a tape file, the block size of the output file defaults to that of the input file. Otherwise, the output file block size defaults to the size used when the tape was mounted.

Acceptable values for block size *n* range from 20 to 65,532. To ensure correct data interchange with other DIGITAL systems, however, specify a block size of not more than 512 bytes. For compatibility with most non-DIGITAL systems, the block size should not exceed 2048 bytes.

/INDEXED_SEQUENTIAL

Output File Qualifier

Defines file organization.

format

output filespec/**INDEXED_SEQUENTIAL**

/OVERLAY

Output File Qualifier

Specifies that the output file is to be overlaid on, or written to, an existing empty file.

format

output filespec/**OVERLAY**

/RELATIVE

Output File Qualifier

Defines output file organization as relative.

format

output filespec/**RELATIVE**

/SEQUENTIAL

Output File Qualifier

Defines output file organization as sequential.

format

output filespec/**SEQUENTIAL**

TFF Facility

1 Using the Terminal Fallback Facility

The VMS Terminal Fallback Facility (TFF) provides table-driven character conversion for terminals. Because every computer terminal can display only one set of characters at a time and each keyboard has a limited number of keys, software developed with one character set or terminal can be impossible to use with another character set or terminal. To help you bridge the gap between incompatible character sets and incompatible terminals, TFF converts characters transparently to software applications (unnoticed by the application software unless explicit inquiries are made). TFF can convert one character to many for characters sent to a terminal, and one to one for characters entered from a terminal.

TFF provides terminal fallback. When an application program sends a character that a terminal cannot display, TFF replaces that character with the closest possible visual character that the terminal can display. This is called *fallback*.

Finally, TFF can perform character compose emulation on input from a terminal. You can create characters that have no associated face on the keyboard by combining two existing characters. This process is known as *composing*. Although TFF offers compose sequence tables, you can also control which keys are *auto-compose* keys.

One of the applications of TFF is to allow users with National Replacement Character (NRC) set terminals to use software developed with the Digital Multinational Character Set (MCS). MCS is essentially the ASCII character set plus 128 characters currently used by owners of NRC terminals around the world.

TFF supersedes the function formerly provided by the (VMS Version 4.x) DCL command SET TERMINAL/FALLBACK. Your system manager uses Terminal Fallback Utility (TFU) commands to set up the TFF environment; you can use TFU commands to set your TFF terminal parameters.

1.1 The Purpose of Terminal Fallback

Terminals have physical limits. Every computer terminal can display one set of characters and each keyboard has a limited number of keys. Characters are arranged into character sets, where each character has a cardinal number (1, 2, 3, and so forth). Computers then use a character's cardinal number to tell the terminal to display that character. For example, to display the character *A*, a computer sends the binary value 64 to the terminal. In the same way, when you press the key labeled *A*, the terminal sends the binary value 64 to the computer.

One common character set is the ASCII character set, designed by the United States primarily for the English language. The ASCII character set, however, does not include many characters used in languages other than English. For example, the ASCII character set does not include accented characters.

Because of the limitations of the ASCII character set, many countries replace some symbols in ASCII with local characters, mostly accented, to produce their own variant of ASCII. A country-specific variant of ASCII is called a National Replacement Character set (NRC).

NRCs do not, however, solve the needs of all countries. Few countries are able to get all the characters they want into ASCII, because ASCII consists of a fixed set of symbols. Also, different countries replace the same ASCII symbol with different local characters. This leaves application software highly dependent on a country's NRC. Software designed using one country's NRC cannot be used in other countries.

In an effort to solve these problems, Digital Equipment Corporation designed a Multinational Character Set (MCS). MCS contains twice as many characters as ASCII, and covers the needs of most European languages. The VT200-series, VT300-series, and VAX workstation terminals, for example, use MCS.

To use an NRC terminal with an MCS-specific application, characters must be converted going to and from the terminal to MCS. The Terminal Fallback Facility provides this conversion transparently to the application through a library of character tables.

Because most NRC terminals cannot display all the MCS characters, TFF replaces those characters with fallback characters. For example, if a terminal cannot display the Japanese yen sign ¥, TFF sends a Y.

1.2 The Purpose of Compose Characters

Sometimes, you cannot use software developed with one character set on a terminal that does not include all of the required characters. Because each terminal keyboard has a limited number of keys, you must use compose sequences to create characters that have no associated face on the keyboard. TFF provides two compose sequence tables, LATIN_1 and ISO_COMPOSE. The default compose sequence table is LATIN_1. You should use this table with DIGITAL applications. However, if an application uses a character set other than MCS, you need a matching compose sequence table. For example, to use the ISO table ISO_VT100MCS, you need the compose sequence table ISO_COMPOSE. Use Terminal Fallback Utility commands to choose TFF tables. After you choose the tables, TFF handles the conversion process.

1.2.1 Composing Characters with TFF

To compose a character in the TFF environment, you press CTRL/K, and then enter the two existing keyboard characters that make up the compose sequence. For example, to compose the copyright sign,©, press CTRL/K followed by CO; to compose ñ, press CTRL/K followed by n~ (lower case n followed by tilde). You can create some characters from more than one compose sequence. Some compose sequences are order or case sensitive, or both. To abort a compose sequence, press the DELETE key. If you press CTRL/K before completing the compose sequence, TFF restarts the compose sequence. If you press any other control key before completing the compose sequence, the compose sequence fails and the control character is sent to the application.

1.2.2 Compose Sequences—DIGITAL LK201 Keyboard and the LATIN_1 Table

Table TFF–1 contains the sequences defined for the standard LK201 keyboard used with VT200-series terminals, DECmate, VAXmate, and DIGITAL workstations.

TFF converts characters that do not have an accurate visual representation to the closest possible fallback representation. If no such fallback exists, TFF replaces the character with an underscore.

Table TFF–1: LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
"	"	quotation mark	" space	
#	#	number sign	+ +	
'	'	apostrophe	' space	
@	@	commercial at	a a	
[	[	opening bracket	((	
\	\	backslash	/ / or /	
]	]	closing bracket	))	
^	^	circumflex accent	^ space	
'	'	single quote	' space	
{	{	opening brace	(-	
|	|	vertical line	/ ^	
}	}	closing brace	) -	
~	~	tilde	~ space	
!	¡	inverted exclamation	! !	
c	¢	cent sign	c / or c |	
L	£	pound sign	l - or l =	
Y	¥	yen sign	y - or y =	

Table TFF-1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
_	§	section sign	s o or s 0 or s !	
_	⊗	currency sign	x o or x 0	
_	©	copyright sign	c o or c 0	
a	ª	female ordinal indicator	a -	
<	«	double open angle brackets	< <	
_	°	degree sign	0 ^	
_	±	plus/minus sign	+ -	
1	¹	superscript 1	1 ^	
2	²	superscript 2	2 ^	
3	³	superscript 3	3 ^	
u	µ	micro sign	/ u	Order
_	¶	paragraph sign (pilcrow)	P !	
.	·	middle dot	^ .	
o	º	masculine ordinal indicator	o _	
>	»	double close angle brackets	> >	
_	¼	fraction one-quarter	1 4	Order
_	½	fraction one-half	1 2	Order
?	¿	inverted question mark	? ?	
A	À	A grave	A `	Case
A	Á	A acute	A '	Case
A	Â	A circumflex	A ^	Case
A	Ã	A tilde	A ~	Case
A	Ä	A umlaut	A "	Case
A	Å	A ring	A *	Case
_	Æ	A E ligature	A E	Order & Case
C	Ç	C cedilla	C ,	Case
E	È	E grave	E `	Case
E	É	E acute	E '	Case
E	Ê	E circumflex	E ^	Case
E	Ë	E umlaut	E "	Case
I	Ì	I grave	I `	Case

Table TFF–1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
I	Í	I acute	I ′	Case
I	Î	I circumflex	I ˆ	Case
I	Ï	I umlaut	I ″	Case
N	Ñ	N tilde	N ~	Case
O	Ò	O grave	O `	Case
O	Ó	O acute	O ′	Case
O	Ô	O circumflex	O ˆ	Case
O	Õ	O tilde	O ~	Case
O	Ö	O umlaut	O ″	Case
O	Ø	O slash	O /	Case
—	Œ	O E ligature	O E	Order & Case
U	Ù	U grave	U `	Case
U	Ú	U acute	U ′	Case
U	Û	U circumflex	U ˆ	Case
U	Ü	U umlaut	U ″	Case
Y	Ÿ	Y umlaut	Y ″	Case
—	ß	German small sharp s	s s	Case
a	à	a grave	a `	Case
a	á	a acute	a ′	Case
a	â	a circumflex	a ˆ	Case
a	ã	a tilde	a ~	Case
a	ä	a umlaut	a ″	Case
a	å	a ring	a *	Case
—	æ	a e ligature	a e	Order & Case
c	ç	c cedilla	c ,	Case
e	è	e grave	e `	Case
e	é	e acute	e ′	Case
e	ê	e circumflex	e ˆ	Case
e	ë	e umlaut	e ″	Case
i	ì	i grave	i `	Case
i	í	i acute	i ′	Case
i	î	i circumflex	i ˆ	Case
i	ï	i umlaut	i ″	Case

Table TFF–1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
n	ñ	n tilde	n ~	Case
o	ò	o grave	o `	Case
o	ó	o acute	o '	Case
o	ô	o circumflex	o ^	Case
o	õ	o tilde	o ~	Case
o	ö	o umlaut	o "	Case
o	ø	o slash	o /	Case
–	œ	o e ligature	o e	Order & Case
u	ù	u grave	u `	Case
u	ú	u acute	u '	Case
u	û	u circumflex	u ^	Case
u	ü	u umlaut	u "	Case
y	ÿ	y umlaut	y "	Case

Note that the characters circumflex (^), tilde (~), and grave accent (`) are used frequently to compose MCS characters. Many NRC sets, however, replace them with an NRC character. Thus, they are not available on an NRC keyboard. For each one of these keys, TFF accepts a replacement key. If you use this replacement key, however, you must begin the compose sequence with it; these compose sequences are order sensitive. For example, you can compose Ê with ^E, E^ or $E, but not with E$. TFF offers the following replacement keys:

- $ (dollar sign) replaces ^ (circumflex)
- % (percent) replaces ~ (tilde)
- & (ampersand) replaces ` (grave accent)

1.3 Setting TFF Terminal Parameters

After your system manager has installed and set up the environment for TFF on your system, you can use Terminal Fallback Utility (TFU) commands to set, change, and display TFF terminal parameters. You can choose and set the default conversion tables for your terminal.

Typically, your system manger sets one fallback table and one compose sequence table as system defaults. These, and perhaps other tables, are loaded into nonpaged dynamic memory pool, making them available to you. If you want to make use of the system conversion table defaults, you can log in to any local terminal and enter a DCL *or* TFU SET TERMINAL/FALLBACK command. In addition, you can use the TFU SET TERMINAL/FALLBACK command to set your own default fallback tables from any tables previously loaded into nonpaged dynamic memory pool. If you need

a table that is available in the TFF library, but not loaded into nonpaged dynamic memory pool, you must ask your system manager to load the table.

After your system manager sets up the TFF environment, you can make full use of the Terminal Fallback Utility SET TERMINAL/FALLBACK command and its many options. For a detailed description of these options, see the TFU Commands section.

Use the following commands to manage TFF terminal parameters:

Command	Use
DIRECTORY	Displays the conversion tables available in the TFF library.
SET TERMINAL/FALLBACK[=option]	Enables or modifies TFF terminal parameters. This is the primary Terminal Fallback Utility command. Use this command to activate the desired behavior at the specified terminal.
SHOW DEFAULT_TABLE	Displays the default fallback character conversion table.
SHOW TABLES	Displays information about loaded conversion tables. This information is helpful before you try to enable TFF terminal parameters.
SHOW TERMINAL/FALLBACK	Displays TFF terminal parameters.

Appendix A

Character Sets

The following tables present the ASCII character set and the DEC Multinational Character Set.

A.1 ASCII Character Set

The ASCII character set consists of the characters shown in the following table. The characters with names are not printable. You can calculate the numeric value of a character by constructing a 2-digit hexadecimal number in which the column position of the character represents the 16 position of the hexadecimal number and the row position of the character represents the units position of the number. For example, an uppercase *A* has the numeric value 41 hexadecimal. String comparisons are made using these values.

Hex Values	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	K	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

ZK-1774-84

A.2 ASCII and DEC Multinational Character Set Tables

Table A–1 represents the ASCII character set (characters with decimal values 0 through 127). The first half of each of the numbered columns identifies the character as you would enter it on a VT300-, VT200-, or VT100-series terminal or as you would see it on a printer (except for the nonprintable characters). The remaining half of each column identifies the character by the binary value of the byte; the value is stated in three radixes—octal, decimal, and hexadecimal. For example, the uppercase letter A has, under ASCII conventions, a storage value of hexadecimal 41 (a bit configuration of 01000001), equivalent to 101 in octal notation and 65 in decimal notation.

Table A–1: Graphical Representation of the ASCII Character Set

ROW	b8 b7 b6 b5 b4 b3 b2 b1	COLUMN 0		COLUMN 1		COLUMN 2		COLUMN 3		COLUMN 4		COLUMN 5		COLUMN 6		COLUMN 7	
		0 0 0 0		0 0 0 1		0 0 1 0		0 0 1 1		0 1 0 0		0 1 0 1		0 1 1 0		0 1 1 1	
0	0 0 0 0	NUL	0 0 0	DLE	20 16 10	SP	40 32 20	0	60 48 30	@	100 64 40	P	120 80 50	`	140 96 60	p	160 112 70
1	0 0 0 1	SOH	1 1 1	DC1 (XON)	21 17 11	!	41 33 21	1	61 49 31	A	101 65 41	Q	121 81 51	a	141 97 61	q	161 113 71
2	0 0 1 0	STX	2 2 2	DC2	22 18 12	"	42 34 22	2	62 50 32	B	102 66 42	R	122 82 52	b	142 98 62	r	162 114 72
3	0 0 1 1	ETX	3 3 3	DC3 (XOFF)	23 19 13	#	43 35 23	3	63 51 33	C	103 67 43	S	123 83 53	c	143 99 63	s	163 115 73
4	0 1 0 0	EOT	4 4 4	DC4	24 20 14	$	44 36 24	4	64 52 34	D	104 68 44	T	124 84 54	d	144 100 64	t	164 116 74
5	0 1 0 1	ENQ	5 5 5	NAK	25 21 15	%	45 37 25	5	65 53 35	E	105 69 45	U	125 85 55	e	145 101 65	u	165 117 75
6	0 1 1 0	ACK	6 6 6	SYN	26 22 16	&	46 38 26	6	66 54 36	F	106 70 46	V	126 86 56	f	146 102 66	v	166 118 76
7	0 1 1 1	BEL	7 7 7	ETB	27 23 17	'	47 39 27	7	67 55 37	G	107 71 47	W	127 87 57	g	147 103 67	w	167 119 77
8	1 0 0 0	BS	10 8 8	CAN	30 24 18	(	50 40 28	8	70 56 38	H	110 72 48	X	130 88 58	h	150 104 68	x	170 120 78
9	1 0 0 1	HT	11 9 9	EM	31 25 19	)	51 41 29	9	71 57 39	I	111 73 49	Y	131 89 59	i	151 105 69	y	171 121 79
10	1 0 1 0	LF	12 10 A	SUB	32 26 1A	*	52 42 2A	:	72 58 3A	J	112 74 4A	Z	132 90 5A	j	152 106 6A	z	172 122 7A
11	1 0 1 1	VT	13 11 B	ESC	33 27 1B	+	53 43 2B	;	73 59 3B	K	113 75 4B	[	133 91 5B	k	153 107 6B	{	173 123 7B
12	1 1 0 0	FF	14 12 C	FS	34 28 1C	,	54 44 2C	<	74 60 3C	L	114 76 4C	\	134 92 5C	l	154 108 6C	|	174 124 7C
13	1 1 0 1	CR	15 13 D	GS	35 29 1D	–	55 45 2D	=	75 61 3D	M	115 77 4D	]	135 93 5D	m	155 109 6D	}	175 125 7D
14	1 1 1 0	SO	16 14 E	RS	36 30 1E	.	56 46 2E	>	76 62 3E	N	116 78 4E	^	136 94 5E	n	156 110 6E	~	176 126 7E
15	1 1 1 1	SI	17 15 F	US	37 31 1F	/	57 47 2F	?	77 63 3F	O	117 79 4F	_	137 95 5F	o	157 111 6F	DEL	177 127 7F

KEY

CHARACTER		
ESC	33	OCTAL
	27	DECIMAL
	1B	HEX

ZK-1752-84

The ASCII character set comprises the first half of the DEC Multinational Character Set. Table A–2 represents the second half of the DEC Multinational Character Set (characters with decimal values 128 through 255). The first half of each of the

numbered columns identifies the character as you would see it on a VT300- or VT200-series terminal or printer (these characters cannot be output on a VT100-series terminal).

Table A–2: Graphical Representation of the DEC Multinational Extension to the ASCII Character Set

8		9		10		11		12		13		14		15		COLUMN	
1 0 0 0		1 0 0 1		1 0 1 0		1 0 1 1		1 1 0 0		1 1 0 1		1 1 1 0		1 1 1 1		b8 b7 BITS b6 b5 b4 b3 b2 b1	ROW
	200 128 B0	DCS	220 144 90		240 160 A0	°	260 176 B0	À	300 192 C0		320 208 D0	à	340 224 E0		360 240 F0	0 0 0 0	0
	201 129 B1	PU1	221 145 91	¡	241 161 A1	±	261 177 B1	Á	301 193 C1	Ñ	321 209 D1	á	341 225 E1	ñ	361 241 F1	0 0 0 1	1
	202 130 B2	PU2	222 146 92	¢	242 162 A2	2	262 178 B2	Â	302 194 C2	Ò	322 210 D2	â	342 226 E2	ò	362 242 F2	0 0 1 0	2
	203 131 83	STS	223 147 93	£	243 163 A3	3	263 179 B3	Ã	303 195 C3	Ó	323 211 D3	ã	343 227 E3	ó	363 243 F3	0 0 1 1	3
IND	204 132 B4	CCH	224 148 94		244 164 A4		264 180 B4	Ä	304 196 C4	Ô	324 212 D4	ä	344 228 E4	ô	364 244 F4	0 1 0 0	4
NEL	205 133 B5	MW	225 149 95	¥	245 165 A5	µ	265 181 B5	Å	305 197 C5	Õ	325 213 D5	å	345 229 E5	õ	365 245 F5	0 1 0 1	5
SSA	206 134 B6	SPA	226 150 96		246 166 A6	¶	266 182 B6	Æ	306 198 C6	Ö	326 214 D6	æ	346 230 E6	ö	366 246 F6	0 1 1 0	6
ESA	207 135 B7	EPA	227 151 97	§	247 167 A7	·	267 183 B7	Ç	307 199 C7	Œ	327 215 D7	ç	347 231 E7	œ	367 247 F7	0 1 1 1	7
HTS	210 136 B8		230 152 9B	¤	250 168 A8		270 184 B8	È	310 200 C8	Ø	330 216 D8	è	350 232 E8	ø	370 248 F8	1 0 0 0	8
HTJ	211 137 B9		231 153 99	©	251 169 A9	1	271 185 B9	É	311 201 C9	Ù	331 217 D9	é	351 233 E9	ù	371 249 F9	1 0 0 1	9
VTS	212 138 8A		232 154 9A	ª	252 170 AA	º	272 186 BA	Ê	312 202 CA	Ú	332 218 DA	ê	352 234 EA	ú	372 250 FA	1 0 1 0	10
PLD	213 139 8B	CSI	233 155 9B	«	253 171 AB	»	273 187 BB	Ë	313 203 CB	Û	333 219 DB	ë	353 235 EB	û	373 251 FB	1 0 1 1	11
PLU	214 140 BC	ST	234 156 9C		254 172 AC	¼	274 188 BC	Ì	314 204 CC	Ü	334 220 DC	ì	354 236 EC	ü	374 252 FC	1 1 0 0	12
RI	215 141 BD	OSC	235 157 9D		255 173 AD	½	275 189 BD	Í	315 205 CD	Ý	335 221 DD	í	355 237 ED	ý	375 253 FD	1 1 0 1	13
SS2	216 142 8E	PM	236 158 9E		256 174 AE		276 190 BE	Î	316 206 CE		336 222 DE	î	356 238 EE		376 254 FE	1 1 1 0	14
SS3	217 143 BF	APC	237 159 9F		257 175 AF	¿	277 191 BF	Ï	317 207 CF	ß	337 223 DF	ï	357 239 EF		377 255 FF	1 1 1 1	15

KEY

CHARACTER		
ESC	33	OCTAL
	27	DECIMAL
	1B	HEX

ZK-1753-84

Appendix B

Expressions

The following table lists data operations and comparisons in order of precedence, beginning with the highest:

Operator	Precedence	Description
+	1	Indicates a positive number
–	1	Indicates a negative number
*	2	Multiplies two numbers
/	2	Divides two numbers
+	3	(1) Adds two numbers (2) Concatenates two character strings
–	3	(1) Subtracts two numbers (2) Subtracts two character strings
.EQS.	4	Tests if two character strings are equal
.GES.	4	Tests if first character string is greater than or equal
.GTS.	4	Tests if first character string is greater than
.LES.	4	Tests if first character string is less than or equal
.LTS.	4	Tests if first character string is less than
.NES.	4	Tests if two character strings are not equal
.EQ.	4	Tests if two numbers are equal
.GE.	4	Tests if first number is greater than or equal to
.GT.	4	Tests if first number is greater than
.LE.	4	Tests if first number is less than or equal to
.LT.	4	Tests if first number is less than
.NE.	4	Tests if two numbers are not equal
.NOT.	5	Logically negates a number
.AND.	6	Combines two numbers with a logical AND
.OR.	7	Combines two numbers with a logical OR

The following tables demonstrate the results of logical operations on a bit-by-bit basis and a number-by-number basis. In logical operations, a character string beginning

with an uppercase or lowercase T or Y is treated as the number 1; a character string beginning with any other character is treated as the number 0. In logical operations, odd numbers are considered true and even numbers and zero are considered false.

Given Bit A	Bit B	Results .NOT A	A .AND. B	A .OR. B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

Given Number A	Number B	Results .NOT A	A .AND. B	A .OR. B
odd	odd	even	odd	odd
odd	even	even	even	odd
even	odd	odd	even	odd
even	even	odd	even	even

Appendix C

Terminal Keys

The following tables present the operating system's interpretation of keys on terminals in the VT300, VT200, and VT100 series.

C.1 VT300 and VT200 Terminal Series

The following table describes how the operating system responds when various keys and control characters are pressed on VT300- or VT200-series terminals. The table assumes that line editing is enabled (the default). (Characters not mentioned in the table are treated as null characters.)

Character	HEX	System Response
CTRL/A	01	Switches between overstrike and insert modes
CTRL/B	02	Recalls previous line
CTRL/C	03	Interrupts current image (image may define alternate CTRL/C action)
CTRL/D	04	Moves cursor left one character
CTRL/E	05	Moves cursor to end of line
CTRL/F	06	Moves cursor right one character
CTRL/H	08	Moves cursor to beginning of line
CTRL/I	09	Horizontal tab
CTRL/J	0A	Deletes previous word
CTRL/M	0D	Line terminator
CTRL/O	0F	Suspends/resumes echoing of output
CTRL/Q	11	Resumes output (see CTRL/S)
CTRL/R	12	Refreshes current line
CTRL/S	13	Suspends output (see CTRL/Q)
CTRL/T	14	Displays process information (must be enabled with SET CONTROL=T command)

Character	HEX	System Response
CTRL/U	15	Deletes characters from cursor to beginning of line
CTRL/V	16	Passes next character or escape sequence to the image without interpreting it as described in this table
CTRL/X	18	Purges type-ahead buffer; if characters are on the current line, deletes characters from cursor to beginning of line
CTRL/Y	19	Interrupts current image
CTRL/Z	1A	Indicates end of file
Data keys	-	Enter appropriate character
<X	-	Deletes previous character
CTRL	-	Modifies another key
CTRL/[(ESC)	1B	Begins escape sequence
CTRL/F5	-	Executes answerback message
DOWN ARROW	-	Repeats current line
F1 (No Scroll)	-	Suspends/resumes output
F5 (Break)	-	Shuts down transmission line
F6 (Interrupt)	-	Interrupts the current image
F10 (Exit)	-	Terminates the current image or command procedure
F12 (Backspace)	08	Moves cursor to beginning of line
F13 (Line Feed)	-	Deletes previous word
F14 (^A)	01	Switches between overstrike and insert modes
LEFT ARROW	-	Moves cursor left one character
PFn	-	Can be defined (see DEFINE/KEY)
RETURN	-	Line terminator
RIGHT ARROW	-	Moves cursor right one character
TAB	-	Horizontal tab
UP ARROW	-	Repeats current line

C.2 VT100 Terminal Series

The following table describes how the operating system responds when various keys and control characters are pressed on VT100-series terminals. The table assumes that line editing is enabled (the default). (Characters not mentioned in the table are treated as null characters.)

Character	HEX	System Response
CTRL/A	01	Switches between overstrike and insert modes
CTRL/B	02	Recalls previous line
CTRL/C	03	Interrupts current image (image may define alternate CTRL/C action)
CTRL/D	04	Moves cursor left one character
CTRL/E	05	Moves cursor to end of line
CTRL/F	06	Moves cursor right one character
CTRL/H	08	Moves cursor to beginning of line
CTRL/I	09	Horizontal tab
CTRL/J	0A	Deletes previous word
CTRL/M	0D	Line terminator
CTRL/O	0F	Suspends/resumes echoing of output
CTRL/Q	11	Resumes output (see CTRL/S)
CTRL/R	12	Refreshes current line
CTRL/S	13	Suspends output (see CTRL/Q)
CTRL/T	14	Displays process information
CTRL/U	15	Deletes characters from cursor to beginning of line
CTRL/V	16	Passes next character or escape sequence to the image without interpreting it as described in this table
CTRL/X	18	Purges type-ahead buffer; if characters are on the current line, deletes characters from cursor to beginning of line
CTRL/Y	19	Interrupts current image
CTRL/Z	1A	Indicates end of file
Data keys	-	Enter appropriate character
Backspace (^H)	08	Moves cursor to beginning of line
BREAK	-	Shuts down transmission line
CTRL	-	Modifies another key
CTRL/BREAK	-	Executes answerback message
DELETE	-	Deletes previous character
DOWN ARROW	-	Repeats current line
ESC	1B	Begins escape sequence
LEFT ARROW	-	Moves cursor left one character
LINE FEED	-	Deletes previous word
NO SCROLL	-	Suspends/resumes output
PFn	-	Can be defined (see DEFINE/KEY)

Character	HEX	System Response
RETURN	-	Line terminator
RIGHT ARROW	-	Moves cursor right one character
TAB	-	Horizontal tab
UP ARROW	-	Repeats current line

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

NOTES

NOTES

Reader's Comments

VMS General User's Manual
AA–LA98A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

Reader's Comments

VMS General User's Manual
AA–LA98A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

What I like best about this manual is _____

_____

What I like least about this manual is _____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

New Manual Survey

VMS General User's Manual
AA-LA98A-TE

Do you like this manual's binding?

Does this manual describe the tasks you most often perform on a VMS system? If not, which topics need additional discussion?

Are the descriptions of the DCL commands and lexical functions in this manual useful? Do they contain enough information?

Do you have a full VMS Version 5.0 documentation set available? When using the General User's Manual, do you often refer to the VMS Version 5.0 documentation set for more detailed information? Which books do you refer to most often?

Please indicate the type of user/reader that you most nearly represent:

- ☐ System manager
- ☐ Experienced programmer
- ☐ Novice programmer
- ☐ Other (please specify) _____

Name _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____